Mendelova univerzita v Brně
Provozně ekonomická fakulta

Návrh GIS aplikace pro platformu Windows Phone 8

Diplomová práce

Vedoucí práce:
Ing. Vít Ondroušek, Ph.D.

Bc. Ondřej Švehla

Brno 2015

Na tomto místě bych rád poděkoval svému vedoucímu diplomové práce Ing. Vítu Ondrouškovi, Ph. D. za odbornou pomoc, konzultace a cenné rady při zhotovení této práce.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Návrh GIS aplikace pro platformu Windows Phone 8** vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou Směrnicí o zveřejňování vysokoškolských závěrečných prací. Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona. Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 20. května 2015 _____

Abstract

This diploma thesis deals with the design and development of a mobile GIS application intended for the Windows Phone 8 platform. The first part of this thesis is dedicated to describing GIS applications which are already available as well as resources and parts of .NET framework used for development on the platform mentioned above. The second part contains a specific design, an architecture and an implementation of the application. Functionality of chosen parts of the developed GIS application is demonstrated through models created according to UP methodology. Several tests of the application are performed using data from the arboretum of Mendel University to evaluate functionality and usability of the designed solution.

Keywords

Windows Phone 8, GIS application, framework .NET, UP methodology.

Abstrakt

Diplomová práce se zabývá problematikou návrhu a vývoje mobilní GIS aplikace pro platformu Windows Phone 8. V první části práce jsou popsány již vyvinuté GIS aplikace, prostředky dostupné k vývoji pro tuto platformu a části frameworku .NET použité pro vývoj GIS aplikace. V druhé části je popsán samotný návrh a implementace aplikace a její architektura. Chování vybraných částí aplikace je znázorněno na modelech dle metodiky UP. Funkčnost aplikace je ověřena na testovacích datech, které byly vytvořeny v arboretu MENDELU.

Klíčová slova

Windows Phone 8, GIS aplikace, framework .NET, metodika UP.

Obsah

Seznam obrázků

Seznam tabulek

1 Seznam zkratek a vybraných pojmů

GIS	Geografický informační systém
CAM	Computer Aided Mapping
CAC	Computer Aided Cartography
SDK	Software Development Kit
Framework	Aplikační rámec – softwarová struktura
Cloud computing (cloud)	Technologie na sdílení dat
Feature	Mapový prvek (bod, linie, polygon)
API	Application Programming Interface
XAML	Extensible Application Markup Language
SLAT	Second Level Address Translation
DEP	Data Execution Prevention
MDF	Master Database File
SDF	Sqlc Database File
T-SQL	Transact-Structured Query Language
REST	Representational state transfer
JSON	JavaScript Object Notation
Serializace	Rozdělení JSON objektu na části
Deserialiazace	Spojení informací do JSON objektu
WPF	Windows Presentation Foundation
Element	Hlavní značka XML souboru
Page	Jedna obrazovka Windows Phone
GCM	Google CloudMessage
Checkpoint	Kontrolní bod

2 Úvod a cíl práce

V roce 2014 vznikl na PEF Mendelu tým, který vyvíjí GIS aplikaci. Tato aplikace je vyvíjena za účelem naplnění poptávky na poli mobilních GIS aplikací napříč všemi platformami mobilních operačních systémů. Autor této diplomové práce řeší problematiku vývoje komplexní GIS aplikace pro platformu MS Windows Phone.

2.1 Úvod

Aby bylo možné pochopit, o jaký typ aplikace se jedná a k čemu se bude aplikace využívat, je nutné nejdříve vysvětlit, co znamená zkratka GIS. Tato zkratka se používá jako označení oboru Geografické informační systémy.

Než ale bude možné vysvětlit, co znamená název Geografické informační systémy, k čemu se tento obor používá a jaká je jeho problematika, je nutné si uvědomit, co znamená pojem informační systém. Dle definice od Cause a Sachvilla je informační systémy soubor hardware a software používaný za účelem získávání, uchovávání, spojování a vyhodnocování informací. Geografický informační systém je informační systém, který se využívá pro práci s prostorovými daty. Prostorovými daty se rozumí data vypovídající o zemském povrchu. Ke geografickým informačním systémům lze přistupovat z několika pohledů. Tyto přístupy se liší v primární funkci systému. Jedná se o přístup:

- Kartografický – zaměřuje se na prezentaci dat
- Databázový (evidenční) – zaměřuje se na zpracování a uchování dat
- Analytický (modelování) – zaměřuje se na analytické prostředky a je využíván například v geologii nebo meteorologii

Vyvíjená aplikace spadá do kartografického přístupu ke GIS. K tomuto přístupu patří CAM (Computer Aided Maping) a CAC (Computer Aided Cartography) systémy. Z důvodu, že pro zobrazování informací bude aplikace používat mapu, je dále detailněji popsán CAM model. Základními mapovými objekty jsou:

- bod – slouží k reprezentaci objektů, které jsou tak malé, že pro jejich zobrazení nelze použít níže uvedené objekty, např. slouží pro vykreslení stromu
- linie – slouží k reprezentaci objektů, které nemají definovanou šířku nebo je pro jejich reprezentaci nevhodné použít plochu – jsou to například silnice
- plocha (polygon) – slouží k reprezentaci rozměrných objektů, jako jsou například budovy nebo jezera

Tyto základní objekty jsou následně sdružovány do tematických vrstev, které jsou vykreslovány do mapy. Tematickou vrstvou se rozumí například bodová vrstva stromy.

Geografické informační systémy se v současné době využívají v řadě odvětvích lidské činnosti. Jedná se například o následující činnosti:

- obrana – využívá se například pro plánování vojenských cvičení
- správa inženýrských sítí – mapování infrastruktury sítě mobilních telefonů
- školství – využití ve výcvikových simulačních projektech

(Úvod do Geografických informačních systémů, 2005)

2.2 Motivace

Hlavní motivací je nabídnout řešení mobilního geografického informačního systému použitelného na současných běžně dostupných platformách Android, iOS a Windows Phone. Snahou je navázání odborné spolupráce s komerční sférou a zaplnění prázdného místa na trhu, které představují mobilní GIS aplikace napříč všemi výše uvedenými platformami. Primární snaha vývojového týmu je směrována na platformu Android a její komerční využití. Aplikace pro platformy iOS a MS Windows Phone jsou zatím vyvíjeny pro interní účely univerzity, ale s vizí i jiného využití.

2.3 Cíl práce

Cílem práce je vytvoření GIS aplikace pro operační systém MS Windows Phone 8. Aplikace bude poskytovat komplexní funkcionalitu danou typem aplikace, bude tedy poskytovat funkcionalitu pro vytváření, modifikování a ukládání informací o vnějším prostředí (vrstvy bodů, linií a polygonů). Tyto informace bude aplikace vhodným způsobem zobrazovat. Aplikace bude také poskytovat funkcionalitu pro práci s klíčovými vlastnostmi vrstev. Dále bude zajišťovat výměnu dat se serverem. Aplikace bude dodržovat zásady multiplatformního vývoje. Bude tedy vytvořena podle již navrženého designu pro platformy Android a iOS a bude respektovat požadavky uvedené zadavatelem. Po vytvoření aplikace bude přikročeno k testování aplikace. Testování bude provedeno pomocí emulace ve vývojovém prostředí a také přímo na mobilním zařízení. Následně budou zhodnoceny dosažené výsledky, zejména výhody a nevýhody zvoleného přístupu.

3 Rešeršní část

Tato kapitola nejprve pojednává o stávajících dostupných GIS aplikacích. Především o aplikaci ArcGIS, která je v současnosti nejpropracovanější aplikací a dále je popsán nově vznikající projekt QField. Následně je rozebírána platforma Windows Phone 8 jak z pohledu architektury a používaných jmenných prostorů, tak i z hlediska použitého SDK a jeho funkcí. V poslední řadě jsou popsány vybrané části frameworku .NET použité pro vývoj GIS aplikace. Jedná se především o části zabývající se lokální databází, síťovou komunikací a tvorbou GUI.

3.1 Stávající GIS aplikace

3.1.1 ArcGIS

ArcGIS od firmy Esri poskytuje uživatelům škálovatelný framework pro implementaci GIS na desktopech, serverech a mobilních zařízeních. V současné době je ArcGIS ve verzi 10.x. Za účelem vývoje kompletního GIS integruje kolekci produktů GIS software. Zahrnuje následující důležité frameworky:

- ArcGIS Online – mapovací platforma pro přístup přes web
- ArcGIS for Desktop – profesionální systém GIS na desktop
- ArcGIS for Server – platforma pro podnikové distribuce

ArcGIS Online
ArcGIS Online je platforma založená na technologii cloud computingu, která umožňuje uživatelům používat, vytvářet a sdílet mapy, scény nebo data. Dále umožňuje přistupovat k různým mapovým podkladům nebo k aplikacím, připraveným k použití. Data, která jsou vytvářena nebo spravována pomocí ArcGIS Online, jsou ukládána do cloudu spravovaného firmou Esri. Tato platforma je využívána pro přístup k ArcGIS z mobilních zařízení. Aplikace vytvořená pomocí této platformy je dostupná k použití na operačních systémech Android, iOS a Windows Phone a má určité požadavky na systém a hardware. Na operační systém Android je aplikace použitelná od verze 2.3.3. Na operační systém iOS od verze 5 a na Windows Phone od verze 7.5. Aplikace má následující doporučené hardwarové požadavky:

- Procesor ARMv7
- 1 GHz procesor
- 512 MB RAM
- Rozlišení obrazovky 800 x 480 px
- Hardware podporující OpenGL ES 2.0
- GPC
- Fotoaparát

(ArcGIS - System requirements, © 2015)
Protože ArcGIS aplikace je vyvinuta pomocí platformy založené na službách poskytovaných technologií cloud computingu, umožňuje aplikace zobrazovat mapy,

které nebyly vytvořeny samotným uživatelem a nejsou tedy autorizovány. Proto aplikace neposkytuje vždy stejnou funkcionalitu. Pokud je používána mapa, která nebyla autorem vytvořena, poskytuje aplikace následující funkcionalitu pro práci s mapou:

- Navigace na mapě pomocí posouvání, přibližování/oddalování a použití záložek
- Výběr různých mapových podkladů
- Hledání míst a adres
- Měření ploch a vzdáleností
- Měnit viditelnost dat
- Zobrazovat současnou pozici (v závislosti na používaném zařízení) s automatickým posunutím mapy
- Dotazovat se na informace uložené v mapě výběrem prvku nebo pomocí předdefinovaných dotazů

Pokud je uživatel autorem mapy, která je používána, umožňuje aplikace kromě výše uvedené funkcionality následující možnosti:

- Zobrazovat jednotlivá menu
- Vytvářet a editovat data – zahrnuje podporu pro sběr dat, založený na šablonách a editaci složitých datových modelů (například domény a subsystémy)

(ArcGIS - Use the app, © 2015)

Pro práci s mapami se aplikace skládá ze dvou oblastí. Z oblasti Map Hub a Map Page. Map Hub je oblast, kde si uživatel může prohlížet mapy a spravovat své připojení k on-line GIS serveru. Uživatel dále může otevřít nedávno používané mapy, vyhledávat obsah na serveru a přepnout své připojení k jinému on-line GIS serveru. Nalezené mapy může otevřít a prohlédnout si detailněji jejich obsah. Snímek obrazovky je uveden na Obr. 1. Map Page je oblast, kde uživatel může zobrazit obsah mapy. Uživatel si může prohlížet informace o prvcích mapy a místech zájmů. Tyto informace může následně editovat. Oblast Map Page je zobrazena na Obr. 2.
(ArcGIS for Windows Phone overview, © 2014)

Obr. 1 Oblast Map Hub

Zdroj: (ArcGIS for Windows Phone overview, © 2014)

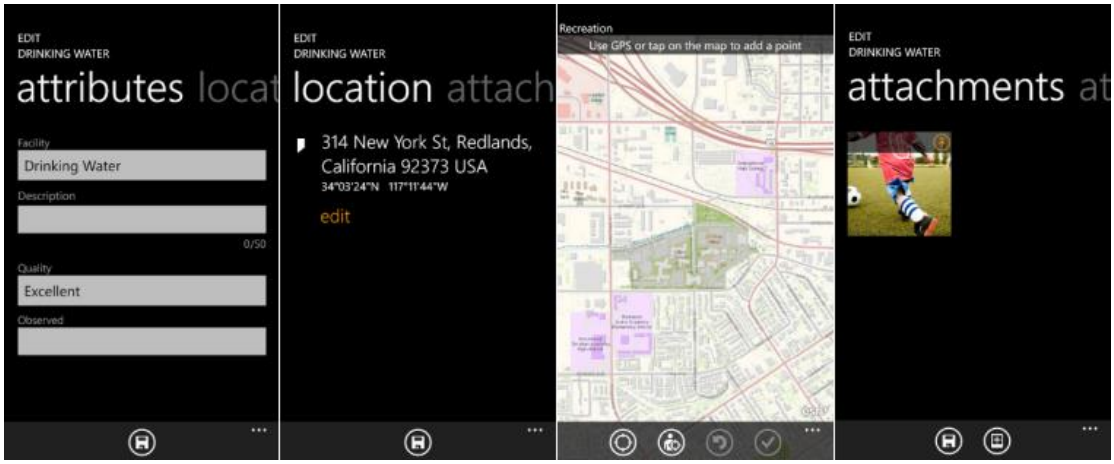
Obrazovka oblasti Map Hub se skládá ze čtyř panelů. Panel „*maps*“ (body 1 – 3) slouží k přístupu k oblíbeným a naposledy používaným mapám. Panel „*account*“ (body 4 – 6) slouží pro správu uživatelského účtu a pro přihlašování do on-line GIS. Panel „*gallery*“ (body 7 a 8) slouží k přístupu a práci s mapami, uloženými v uživatelově on-line GIS. Panel „*settings*“ (body 9 – 12) slouží pro správu připojení k serveru a pro přístup do nastavení aplikace. Tato obrazovka je tedy používána vždy, když uživatel potřebuje změnit jakákoliv natavení nebo používané mapy. (ArcGIS for Windows Phone overview, © 2014)

Obr. 2 Oblast Map Page

Zdroj: (ArcGIS for Windows Phone overview, © 2014)

Obrazovka oblasti Map Page je rozdělena do tří části. V první části (bod 1) je zobrazen pouze titulek mapy. V druhé části (body 2 – 4) se nachází prostor pro zobrazení mapy, zahrnující mapový podklad a operační vrstvy. Dále se uživatel může po mapě navigovat, prohlížet oblasti zájmů a pracovat s vytvořenými prvky. Také může zobrazovat informace o určitém prvku (bod 3) a to pomocí výběru prvku (bod 4). V třetí části bod 5 je umístěna pracovní lišta, obsahující prvky pro práci s mapou. Pracovní lišta je rozdělena do dvou oblastí. První oblast (body 6 – 9) obsahuje prvky pro běžnou práci s mapou. Uživatel může pomocí bodu 6 zobrazit na mapě svou pozici. Pomocí bodu 7 vyhledat na mapě zajímavá místa. Použitím prvku „*tools*“ (bod 8) může uživatel zobrazit menu s nástroji pro měření vzdálenosti, seskupování vytvořených prvků nebo práci se záložkami. Prvek „*contest*“ (bod 9) uživateli poskytuje funkcionalitu pro práci s vytvořenými prvky a detaily mapy. Uživatel může upravovat vlastnosti vrstev, jako jsou například jejich viditelnost. Dále může měnit například mapové podklady, zobrazit legendu mapy apod. Druhá oblast (body 10 – 14) poskytuje obecné funkce pro práci s mapou. Uživatel si zde může například mapu přidat do oblíbených položek nebo mapu kompletně vyčistit.
(ArcGIS for Windows Phone overview, © 2014)

Jednou z nejpoužívanějších funkcí pro práci s mapou je přidávání nových mapových prvků (např. polygon). Pokud chce uživatel přidat nový mapový prvek, bude pro práci s mapou využívat oblast Map Page zobrazenou na Obr. 2 a použije tlačítko tools (bod 8). Poté se uživateli zobrazí pop-up menu ve kterém uživatel zvolí možnost „collect feature“. Následně se uživateli zobrazí obrazovka pro výběr typu mapového prvku v závislosti na typu editovatelné vrstev. Po výběru typu mapového prvku je uživatelovi zobrazena pivot obrazovka (podpora gesta swipe), pomocí které může editovat atributy, umístění a přílohy daného mapového prvku. Obrazovka je zobrazena na Obr. 3. Uživatelský prvek může být umístěn na aktuální pozici uživatele nebo na pozici, kterou zvolí kliknutím do mapy. Lokalizace prvku musí být následně uživatelem potvrzena. Dále uživatel může zvolit přílohy. Ty mohou být různých typů, například obrázek, pdf nebo word dokument, apod.
(ArcGIS - Add a new feature, © 2014)



Obr. 3 Nastavení vlastností uživatelského prvku
Zdroj: (ArcGIS - Add a new feature, © 2014)

Mezi výhody aplikace patří velké množství nastavení aplikace, množství poskytované funkcionality (nejen pro práci s mapou) a kompatibilita s ostatními frameworky balíku ArcGIS. Hlavní nevýhodou aplikace je těžká ovladatelnost aplikace, čímž se stává pro použití v terénu těžko použitelnou.

3.1.2 QField

Je nově vznikající projekt firmy OPENGIS.ch. Tento projekt vznikl v roce 2014 založením na dřívějším projektu QGIS mobile. QFiled umožňuje uživateli používat projekty vytvořené desktopovým programem QGIS na mobilním zařízení. Aplikace je kompletně vyvinuta s myšlenkou jednoduché a pohodlné ovladatelnosti v terénu. Za tímto účelem implementuje následující vzory:

- GPS centric – mapa je centrována podle aktuální pozice uživatele
- Práce v offline režimu bez jakéhokoliv omezení
- Synchronizace
- Optimalizované GUI pro použití v terénu
- Přepínání mezi různými módy – měření, zobrazování map, apod.

V současné době poskytuje aplikace funkce umožňující navigaci na mapě pomocí posouvání a přibližování/oddalování mapy. Dále identifikovat vytvářené mapové prvky a následně je editovat.

Mezi výhody této aplikace patří dobrá ovladatelnost při práci v terénu. Hlavní nevýhodou aplikace je omezená funkcionalita (zaviněno aktuálním vývojem aplikace).
(QField, © 2015)

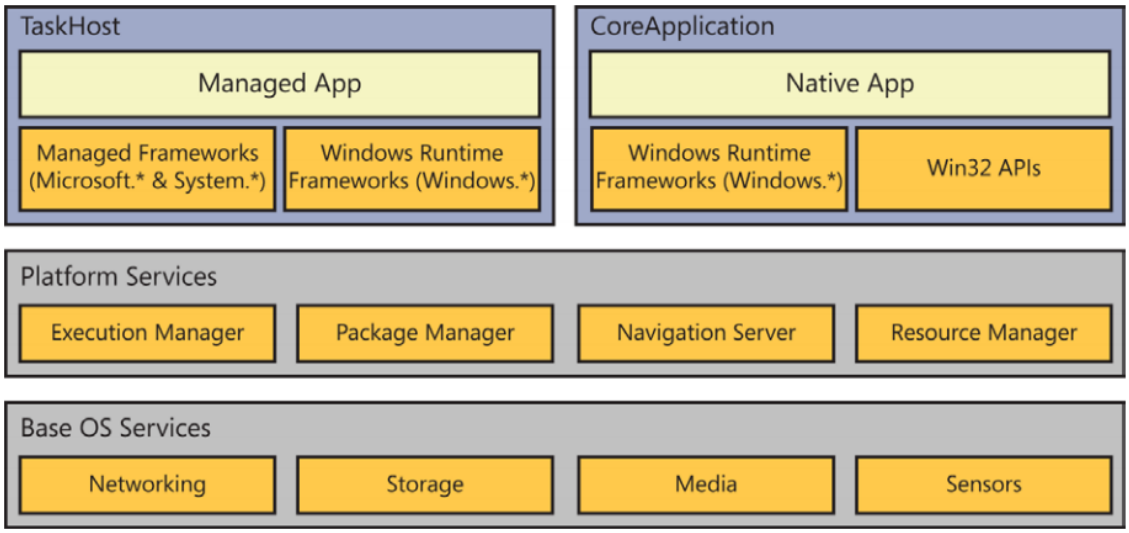
3.2 Platforma Microsoft Windows Phone 8 z pohledu vývojáře

3.2.1 Architektura Windows Phone 8

Architektura platformy Windows Phone 8 je založena na přístupu ke dvěma zcela odlišným modelům, představujících aplikační vrstvu. Tyto modely jsou rozděleny podle technologie použité pro vytvoření uživatelského rozhraní. Jedná se tedy o model podle technologie XAML a model podle technologie Direct 3D. Architektura platformy je znázorněna na Obr. 4. Model XAML je reprezentován blokem „*TaskHost*“, který byl hlavním modelem pro vývoj aplikací ve verzi Windows Phone 7 a technologie XAML je popsána níže v kapitole 3.3.3. Pokud je tedy aplikace vyvíjena podle tohoto modelu, je zajištěna zpětná kompatibilita mezi verzemi platformy 8.x a 7.x. Pokud je aplikace vyvíjena podle modelu Direct3D reprezentovaného blokem „*CoreAppliaciton*“, tak je aplikace kompatibilní pouze s verzí platformy 8 a výše. Oba modely využívají společné platformní služby, pomocí kterých přistupují k základní vrstvě operačního systému (senzory, media, uložiště, apod.). Jedná se o následující platformní služby:

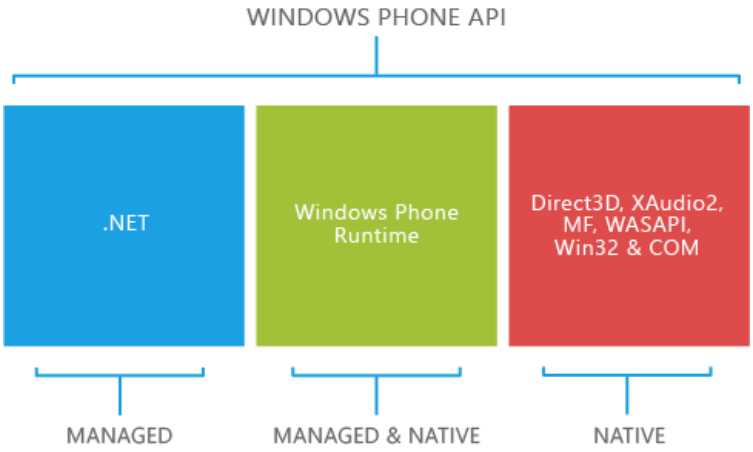
- *Package Manager* slouží k instalaci a odinstalaci aplikace a zachování všech metadat (data o aplikaci, např. přístupová práva nebo číslo verze) po celou dobu životního cyklu aplikace. Dále zajišťuje zachování případného připnutí aplikace jako dlaždici na úvodní obrazovku nebo dalších přístupových bodech systému.
- *Execution Manager* zajišťuje veškerou logiku aplikace. Obsluhuje procesy při běhu aplikace, provádí úkoly na pozadí nebo plánuje provádění úkolů.
- *Navigation Server* slouží ke správě navigace aplikace při jejím běhu. To znamená, že zajišťuje přechody mezi jednotlivými obrazovkami aplikace nebo přepínání mezi spuštěnými aplikacemi.
- *Resource Manager* zajišťuje správu systémových zdrojů telefonu. Přiděluje aplikaci čas na procesoru telefonu nebo paměťové prostředky.

(MCKEANA, 2013)



Obr. 4 Architektura platformy Windows Phone 8
Zdroj: (MCKEANA, 2013)

Výše popisované modely se liší nejen v technologie použité pro tvorbu GUI, ale i v přístupu k Windows Phone API. Windows Phone API používá tři hlavní skupiny API a to `.NET API`, `Windows Phone Runtime` a `Win32 a COM API`, jak je uvedeno na Obr. 5. K vývoji aplikací dle pomocí XAML se používají `.NET API` nebo `Windows Phone Runtime`. Pro vývoj aplikací dle Direct3D se využívají `Windows Phone Runtime` nebo `Win32 a COM API`. Ve všech skupinách API je umožněn vývoj jazycích C# nebo Visual Basic .NET (VB.NET).
(MCKEANA, 2013) (Windows Phone API reference, © 2015)



Obr. 5 Hlavní jmenné prostory Windows Phone API
Zdroj: (Windows Phone API reference, © 2015)

.Net API
Jedná se o API, které zajišťuje zpětnou kompatibilitu s verzí systému Windows Phone 7.x. Obsahuje jmenné prostory `System` a `Microsft.Phone`. Mezi zajímavé knihovny z pohledu vývoje GIS aplikaci patří:

- `Microsoft.Devices` – jmenný prostor pro přístup k hardware
- `Microsoft.Phone.Maps` – jmenný prostor pro práci s mapami
- `System.Device.Location` – umožňuje přístup k GPS a zjištění aktuální polohy
- `System.IO.IsolatedStorage` – umožňuje práci s virtuálním souborovým systémem

(.NET API for Windows Phone, © 2015)

Windows Phone Runtime
`Windows Phone` Runtime je podmnožinou nativního rozhraní `WinRT`, které se používá pro vývoj desktopových aplikací pro OS Windows 8. Manažované jazyky jsou doplněny o nativní jazyk C++. Zajímavými jmennými prostory jsou:

- `Windows.Devices.Geolocation` – slouží k získání GPS souřadnic
- `Windows.Phone.Speech` – rozpoznávání a syntéza řeči

(Windows Phone Runtime API, © 2015)

Win32 a COM API
API napsané v nativním kódu C++, které umožňuje přenos stávajících aplikací z Windows 8 používajících `Win32 API` na mobilní zařízení s platformou Windows Phone 8. Dále umožňuje využívat jmenné prostory pro vývoj her. Také obsahuje nativní API pro práci s kamerou nebo nahrávání zvuku.
(Win32 and COM API, © 2015)

3.2.2 Windows Phone SDK 8.0

Windows Phone SDK 8.0 je doplňkový balík pro vývojové prostředí, kterému přináší rozšiřující funkcionalitu pro umožňující vývoj pro platformu Windows Phone verze 8 a 7.5. Toto SDK obsahuje nástroje, jakou jsou např. nástroj Blend pro návrh prvku uživatelského rozhraní nebo animací aplikace, emulátor umožňující testování aplikace nebo aplikace ISETool pro simulaci isolovaného uložiště telefonu.

Emulátor
Windows Phone Emulátor je desktopová aplikace, prostřednictvím které lze nahradit fyzické mobilní zařízení se systémem Windows. Emulátor poskytuje virtualizované prostředí, pomocí kterého je možné jednoduchým způsobem ladit a testovat vyvíjenou aplikaci. Architektura a výkon Windows Phone Emulátoru jsou navrženy tak, aby emulátor poskytoval výkon a funkce srovnatelné s fyzickým zařízením.

Emulátor je používán jako virtuální stroj pracující na technologii Hyper-V, což je virtualizační technologie pro operační systém Windows 8. Právě díky nutnosti použití technologie Hyper-V není možné spustit emulátor z jiného operačního systému. Aby mohl být emulátor spuštěn, musí operační systém splňovat následující požadavky:

- podpora hardwarové asistence vritualizace
- podpora technologie Second Level Address Translation (SLAT)
- podpora technologie Data Execution Prevention (DEP)
- 4 GB paměti RAM
- 64-bitovou verzi operačního systému Windows 8 a výše
- podpora technologie DHCP
- automaticky konfigurované nastavení DNS a brány

(System requirements for the emulator for Windows Phone 8, © 2015)

Jak bylo zmíněno výše, pomocí emulátoru lze téměř plnohodnotně ladit aplikaci bez použití fyzického zařízení. Jednou z důležitých funkcí emulátoru je testování zobrazování aplikace při různých rozlišeních obrazovky. Emulátor umožňuje používat rozlišení:

- WVGA (800 x 480)
- WXGA (1280 x 768)
- HD rozlišení 720p (1280 x 720))

Další důležitou funkcí je testování výkonosti při omezené velikosti RAM paměti. Emulátor umožňuje testování aplikací pro platformu Windows Phone 8 s velikostí RAM paměti 512MB a pro platformu Windows Phone 7.x s velikostmi paměti 512MB a 256MB. Dalšími důležitými funkcemi emulátoru při vývoji GIS aplikace je simulace GPS lokalizace a akcelerometru. GPS lokalizaci lze simulovat pomocí vytvoření bodu na mapě (záložka „*Location*“ v menu „*Additional Tools*“) představující aktuální polohu zařízení nebo seznamem bodů, který představuje trasu pohybu zařízení. Dalšími užitečnými funkcemi je možnost testování fotoaparátu nebo mikrofonu. Pomocí Windows Pohne Emulátoru, avšak nelze simulovat funkce, jako jsou například kompas nebo gyroskop.
(Windows Phone Emulator for Windows Phone 8, © 2015)

3.3 Části frameworku .NET relevantní pro návrh GIS aplikace

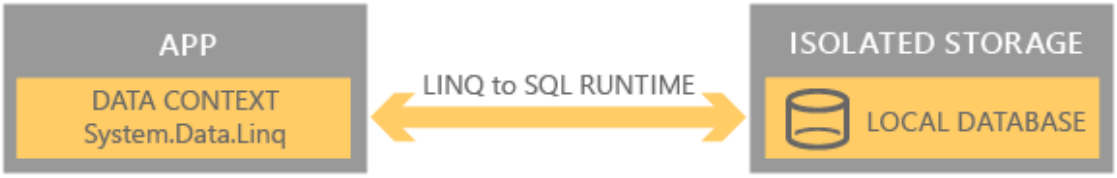
3.3.1 Tvorba lokální databáze

Aby aplikace byla schopná pracovat s daty, je nutné tyto data ukládat. K ukládání dat existuje řada nejrůznějších přístupů, kterými jsou např. ukládání do strukturovaných souborů (např. XML) nebo do lokálních databází. Ukládání dat do lokálních databází je zřejmě nejmodernějším způsobem. Lokální databáze jsou vytvářeny pomocí Microsoft SQL Serveru a jsou reprezentovány typy souborů `*.mdf` (Master Database File), resp. `*.sdf` (Sqlc Database File). K ukládání dat

na mobilních zařízeních se používá typ `*.sdf`. Tyto soubory obsahují jednotlivé tabulky s nadefinovanými relacemi. Pro přístup k databázím se používají technologie, jako jsou např. LINQ to SQL nebo vestavěné procedury na serveru. Pro přístup k databázi lze v rámci vývoje aplikace pro platformu Windows Phone použít pouze technologii LINQ to SQL, která je ale zároveň nejmodernějším přístupem. Proto je tato technologie níže detailněji rozebírána.

LINQ to SQL

LINQ to SQL je používán k definování schématu databáze, k výběru a ukládání nových nebo modifikovaných dat. Tvoří rozhraní mezi lokální databází a aplikací, reprezentovanou určitým datovým kontextem.



Obr. 6 Vztah LINQ to SQL k aplikaci
Zdroj: (Local database for Windows Phone 8, © 2015)

Při položení dotazu na databázi LINQ to SQL překládá integrované dotazy jazyku (speciální syntaxe dotazu `select` apod.) do T-SQL dotazů a následně zasílá do databáze k provedení. Odpověď databáze je následně přeložena z T-SQL do databázového objektu. Aby LINQ to SQL mohl aplikaci poskytovat prostředky pro komunikaci s databází, musí mít aplikace určitý datový kontext. Datový kontext je vytvořen pomocí třídy `DataContext` zahrnuté do knihovny `System.Data.Linq`.

Datový kontext je zástupce objektu, který reprezentuje databázi. Při jeho vytváření je nutné uvést připojovací řetězec k databázi. Pomocí připojovacího řetězce lze také určit, kam se bude databáze v zařízení ukládat. Uvedením klíčového slova `isostore` se bude databáze ukládat do lokální složky uložiště telefonu. Při uvedení klíčového slova `appdata` do instalační složky aplikace. Datový kontext obsahuje tabulky objektů, z nichž každá reprezentuje tabulku v databázi. Každý tabulkový objekt se skládá z entit, které korespondují s řádky dat v databázi. Atributy každé entity popisují strukturu databázové tabulky a definují mapování mezi objektovým modelem dat a schématem databáze. Mapovací atributy určují specifické funkce objektů. Tyto atributy jsou znázorněny v Tab. 1.

Tab. 1 Mapovací atributy LINQ to SQL

Atribut	Popis
`Table`	Mapuje datovou třídu na tabulku v databázi
`Column`	Mapuje atribut datové třídy na sloupec mapované tabulky
`Index`	Mapování indexů pro přístup k sloupcům
`Association`	Mapování relací mezi datovými třídami, typicky se tak takto vytváří cizí klíče

Základní vlastnost, která se používá u atributů `Table` a `Colomun` je `Name`, která specifikuje název tabulky nebo sloupce. Při mapování atributů datové třídy na sloupce tabulky lze uvádět vlastnosti upravující funkcionalitu mapování. Například uvedením vlastnosti `IsPrimaryKey = true` lze označit vlastnost datové třídy jako primární klíč určité tabulky. Dále uvedení například `IsDbGenerated` lze určit automatické generování jedinečné hodnoty primárního klíče.

Velmi důležitou oblastí, při mapování datových tříd na tabulky databáze je vytváření relací. V LINQ to SQL jsou relace vytvářeny obdobným způsobem, jako ve standardních databázových systémech. To znamená, že každá datová třída musí obsahovat atribut mapovaný na cizí klíč v databázi. K vytvoření relace jsou používány následující třídy a atributy:

- Kolekce `EntitySet` – používá se k přístupu k dceřiným datovým třídám
- Kolekce `EntityRef` – používá se k přístupu k rodičovské datové třídě
- Atribut `Association` – je aplikován na kolekci `EntitySet` nebo `EntityRef` a definuje spojení mezi datovými třídami

Spojení mezi datovými třídami pomocí atributu `Association` je vytvořeno uvedením následujících vlastností atributu:

- `Storage` – umožňuje přistoupit ke kolekci `EntitySet` nebo `EntityRef` a uložit do ní asociovanou referenci na datovou třídu
- `OtherKey` – představuje primární klíč v asociované datové třídě
- `ThisKey` – představuje cizí klíč v aktuální datové třídě

Operace nad lokální databází se provádí pomocí práce s mapovanými datovými třídami. Jednotlivé operace se provádí buď pomocí speciálních příkazů, nebo pomocí předdefinovaných metod. Operace vyhledání (`select`) se provádí pomocí následujícího příkazu:

```
from [mapovaný objekt] in [tabulka] where [podmínka] select [atributy]
```

Operace vložení (`insert`) se provádí pomocí předdefinované metody `InserOnSubmit()`, kdy se do mapované tabulky vloží mapovaný objekt. Vložení se musí potvrdit metodou `SubmintChanges()`.

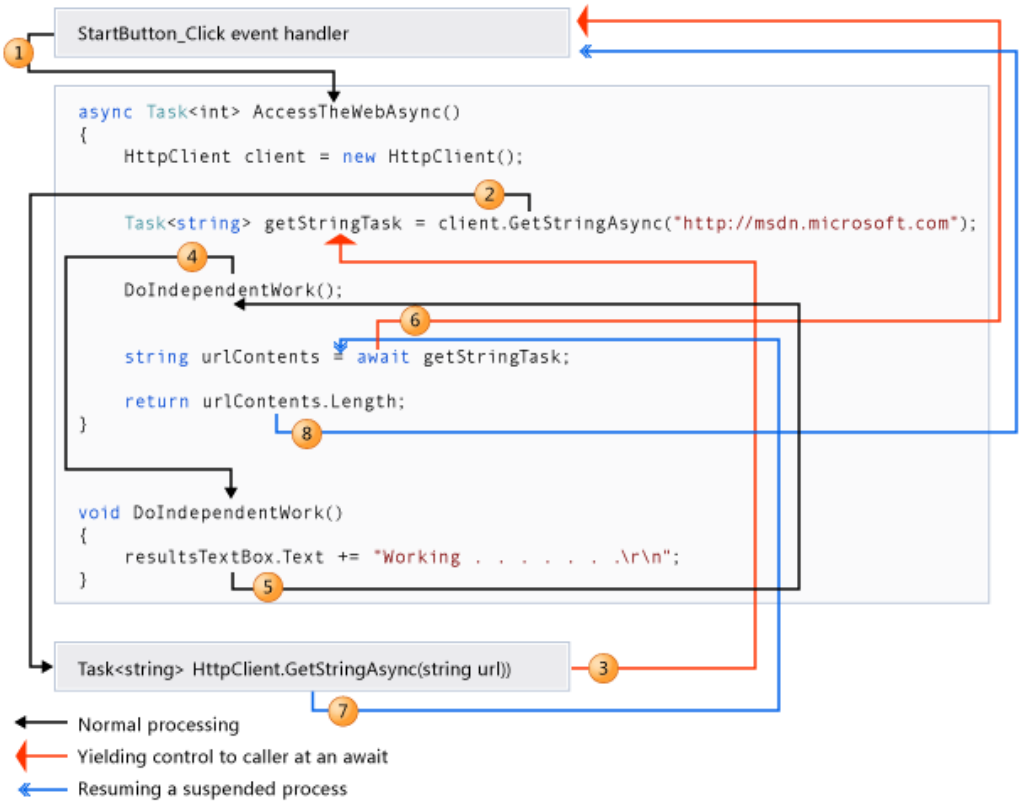
Operace modifikace (`update`) se provádí pomocí operace vyhledání. Ve vyhledaném objektu se následně modifikují vlastnosti. Modifikace se následně musí potvrdit metodou `SubmintChanges()`.

Při operace smazání (`delete`) se nejdříve musí pomocí operace vyhledání vyhledat v mapované tabulce objekt ke smazání. Následně se pomocí předdefinované metody `DeleteOnSubmit()` vymaže objekt z databáze. Vymazání objektu se musí potvrdit metodou `SubmintChanges()`.
(Local database for Windows Phone 8, © 2015)

3.3.2 Podpora HTTP komunikace

V současné době je velmi populární sdílení dat. Sdílení dat probíhá pomocí nahrávání dat do cloudu nebo na servery s databází. Aby ale mohlo být ke sdílení přikročeno, musí aplikace využívat nějaký HTTP klient pro navázání komunikace. Více o HTTP komunikaci níže. Dalším významným problémem je struktura dat pro přenos. Existuje mnoho způsobů jak data pro přenos strukturovat, např. jako text nebo strukturovaný soubor JSON. Přeprava dat pomocí JSON souborů je v současné době velmi využívána a proto je níže rozebírána.

HTTP komunikace
Jedním z možných způsobů jak vytvořit HTTP klient je použití stejně jmenované třídy `HttpClient`. Tato třída je součástí .Net Frameworku ve verzi 4.5 . Třída poskytuje velmi snadný způsob k připojení ke službám přes internet pomocí základních REST služeb. Vytvoření dotazů použitých pro komunikaci pomocí třídy `HttpClient` probíhá použitím metod, které jsou označeny stejnými slovy jako metody HTTP protokolu. Jedná se tedy o metody `GET`, `POST`, `PUT` a `DELETE`. Tyto metody jsou asynchronní. To znamená, že provádění těchto metod probíhá ve vlastním vlákně souběžně s hlavním vláknem programu. Metoda, ve které jsou výše uvedené metody volány, se musí označit klíčovým slovem `async`. Následně na místě, kde je očekáváno využití odpovědi na HTTP dotaz, je nutné použít klíčové slovo `await`. Díky tomu je určen bod, který je použit jako návratové místo po ukončení vlákna s metodou HTTP komunikace. Rozdíly mezi synchronní a asynchronní komunikací jsou zobrazeny na Obr. 7.
(HttpClient Class, © 2015)



Obr. 7 Rozdíl mezi synchronní a asynchronní komunikací
Zdroj: (Asynchronous Programming, © 2015)

Zpracování JSON souborů

Pro zpracování JSON souborů lze v .NET použít několik technologií, například zpracování pomocí javascript serializátoru, třídy `Json` z knihovny `Windows.Data.Json` nebo využitím externí knihovny `NewtonSoft`. Nejlepších výsledků z hlediska časové náročnosti dosahuje externí knihovna `NewtonSoft`, proto bude následně detailněji popsána .

Zpracováním JSON souborů se rozumí jejich serializace a deserialiazace. Serializace a deserializace lze dosáhnout dvěma způsoby.

Prvním způsobem je zpracování pomocí třídy `JsonConvert`. K těmto účelům má třída `JsonConvert` statické metody `SerializeObject<>()` a `DeserializeObject<>()`. Tyto metody jsou generické, umožňují jednoduchou konverzi mezi programátorem vytvořenými třídami a JSON soubory. Tento způsob lze použít pouze v tom případě, že programátorem vytvořená třída obsahuje vlastnosti standardních datových typů, jakou jsou např. `string`, `int`, `array` apod. Pokud je možné tento způsob použít, udává se název třídy, na kterou chceme JSON konvertovat do ostrých závorek. Jako parametr se těmto metodám předává JSON objekt v typu `string`

Druhým způsobem je využití přístupu LINQ to JSON téže knihovny. Tento přístup je založen na využití předdefinovaných objektů `JObject`, `JArray` a `JValue`. LINQ to JSON se používá v případě, že programátorem vytvořené třídy obsahují nestandardní vlastnosti, např. `List`, `Dictionary`, jiné vlastní objekty apod. Každý objekt obsahuje statickou metody `Parse()`, které se jako parametr předává JSON objekt v typu `string`. Následně je nutné manuálně konvertovat vlastnosti mezi vlastními objekty a předdefinovanými objekty z LINQ to JSON. (Json.NET Documentation, © 2015)

3.3.3 Tvorba GUI

GUI mobilních aplikací je vytvářeno propojováním jednotlivých obrazovek. V platformě Windows Phone se tyto obrazovky nazývají „*pages*“ (dále stránky). Pro tvorbu stránek se používá technologie Windows Presentation Foundation, zkráceně WPF. WPF odděluje vzhled uživatelského rozhraní od jeho chování. Vzhled GUI je obvykle vytvářen pomocí Extensible Application Markup Language (XAML) a chování je realizováno ve specifickém jazyce, např. C#. Obě části jsou spojovány pomocí událostí, příkazů a vázání dat.

XAML je jednoduchý značkovací jazyk založený na XML k vytváření a inicializaci objektů v hierarchické struktuře. Pomocí XAML lze obecně vytvořit jakýkoliv strom objektů. Struktura XAML souboru je tvořena stromovou hierarchií jednotlivých elementů. Objekt element se používá pro deklaraci instance nějakého grafického prvku. Z hlediska syntaxe musí element splňovat následující strukturu `<typeName attibutes>`. Část elementu `type-Name` představuje název grafického prvku, od kterého chceme vytvořit instanci. Dále může následovat posloupnost atributů. Příklad definice elementu je znázorněn na Obr. 8

```
<StackPanel>
  <Button Content="Click Me"/>
</StackPanel>
```

Obr. 8 Definice elementu v XAML
Zdroj: (XAML Overview (WPF), © 2015)

Další důležitou součástí XAML je uvádění atributů u jednotlivých prvků. Uvedením atributu lze upravovat vlastnosti grafických prvků, jako jsou např. velikost, název apod. Dále se pomocí atributů stanovuje, jakým událostem bude grafický prvkem naslouchat. Příklad atributu je uveden na Obr. 8 jako vlastnost `Content`.

Některé vlastnosti neleze uvést přímo při definici elementu. Tyto vlastnosti musí být uvedeny do těla elementu se syntaxí pro zápis vlastnosti. Nejprve musí být uveden otevírací tag `<typeName.propertyName>` kde `typeName` představuje typ elementu a `propertyName` název vlastnosti. Dále je uvedena hodnota vlastnosti a ukončovací tag `</typeName.propertyName>` se stejným významem jako otevírací tag.
(XAML Overview (WPF), © 2015)
Protože XAML je jazyk založený na XML, tak jako název elementu lze uvést prakticky jakýkoliv název s neurčitým významem. Aby měly názvy elementů význam

definice grafických prvků, je nutné nadefinovat kontext kódu v jazyce XAML. Kontext XAML kódu je určen uvedením XAML jmenného prostoru.

XAML jmenný prostor je rozšířením XML jmenného prostoru. Způsob definování XAML jmenného prostoru je tedy omezen syntaxí XML jmenného prostoru. Jmenný prostor je definován uvedením URI jako jeho jedinečného identifikátoru. Definice jmenného prostoru je uvedena v kořenové značce (typicky značka `page`) a dědí ji všechny podřízené elementy. Definice jmenného prostoru je vztažena k atributu `xmlns` (XML name space) a skládá se ze dvou deklarací jmenného prostoru. První deklarace představuje výchozí jmenný prostor pro celý WPF klient nebo framework XAML a má následující syntaxi:

```
xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
```

Poté je přistoupeno k druhé deklaraci, která představuje samotnou deklaraci XAML jmenného prostoru a má následující syntaxi:

```
xmlns:x=http://schemas.microsoft.com/winfx/2006/xaml
```

Rozdíl mezi výše uvedenými deklaracemi je ten, že přiřazení prefixu `x:` podporuje vlastní podstatu toho, že takto označené položky jsou součástí definice jazyka XAML a WPF je konkrétní implementace, která využívá XAML jako jazyk ke svému vytváření a definuje slovník elementů pro XAML. Protože je WPF slovník hierarchicky nadřazený slovníku XAML, je mapován jako první.

Po nadefinování XAML jmenného prostoru je nutné provést deklaraci, která označuje slovník standardních prvků GUI. Deklarace je provedena uvedením vlastnosti `:phone` u atributu `xmlns` , která má následující syntaxi, kde `clr-namespace` obsahuje název knihovny se elementy (prvky) a `assembly` obsahuje název kořenové knihovny:

```
xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
```

V neposlední řadě je nutné uvést atribut, který propojí XAML stránku se souborem obsahujícím funkcionalitu pro danou stránku. Jedná se o atribut `x:Class`, kde se se jako URI nastavuje cesta k souboru.
(XAML Namespaces, © 2015)

4 Metodika

Aby bylo možné dosáhnout cílů stanovených v kapitole 2.3 Cíl práce, musí být postupně úspěšně realizovány následující kroky:

- Rozvaha nad požadavky na aplikaci
- Rozvaha nad použitými technologiemi
- Implementace funkční vrstvy
  - Implementace lokální databáze
  - Implementace rozhraní pro komunikaci se serverem
  - Implementace funkčních objektů
- Implementace GUI aplikace
- Testování aplikace

První krok „Rozvaha nad požadavky na aplikaci“ bude zahrnovat detailní nastudování funkčních i nefunkčních požadavků. Nastudovány musí být jak požadavky od zadavatelské firmy, tak i požadavky vyplívající z multiplatformního vývoje. Tento krok je důležitý zejména z důvodu pochopení přesné funkcionality aplikace, díky čemuž se v druhém kroku „Rozvaha nad použitými technologiemi“ lze detailně zaměřit na jednotlivé technologické oblasti důležité pro vývoj.

Ve druhém kroku bude přistoupeno k rozvaze nad použitými částmi frameworku .NET pro vývoj aplikace. Postupně budou rozebrány především třídy, které budou později použity k implementaci lokální databáze a rozhraní pro komunikaci s databází, k implementaci rozhraní pro komunikaci se serverem a také technologie používané pro vytváření GUI.

Dále bude ve třetím kroku nejdříve přikročeno k implementaci lokální databáze. Databáze bude vytvořena pomocí formátu SDF, který jako jediný lze použít pro vytvoření databáze v lokálním uložišti telefonu. Databáze bude vytvořena podle předlohy JSON objektů. To znamená, že názvy objektů budou použity pro pojmenování tabulek, vlastnosti objektů pro nadefinování atributů tabulek a vztahy mezi objekty pro vytvoření relací mezi tabulkami. Podle vhodně zvolené technologie bude následně vytvořeno rozhraní pro komunikaci s databází.

Následně bude přistoupeno k implementaci rozhraní pro komunikaci se serverem. Komunikace bude vytvořena pomocí vhodných tříd umožňujících komunikaci pomocí protokolu HTTP. Rozhraní bude poskytovat metody určené pro čtení, vytvoření, opravu, odstranění.

V dalším kroku bude přistoupeno k vytvoření funkční vrstvy. Postupně tedy budou vytvořeny třídy reprezentující projekt, vrstvu projektu, vrstvu, atributy vrstvy, mapový prvek a atributy mapového prvku. Tyto třídy budou obsahovat stejné vlastnosti jako JSON objekty. Následně bude funkční vrstva doplněna o rozhraní pro komunikaci s databází a pro komunikaci se serverem. Dále bude doplněna o třídu, která bude reprezentovat bod na zemském povrchu, a která bude obsahovat vlastnosti, jako jsou: zeměpisná délka a šířka, přesnost, se kterou byla data naměřena a nadmořská výška. Tuto třídy bude využívat třída pro reprezentaci mapového prvku namísto určitých vlastností z JSON objektu. Následně bude

přikročeno k propojení výše uvedených tříd s rozhraním databáze. To především z toho důvodu, aby byla zajištěna perzistence mezi daty v databázi a daty programu. Dále budou výše uvedené třídy propojeny s rozhraním zajišťujícím komunikaci se server. Budou naimplementovány metody pro serializaci a deserializaci JSON objektů.

V následujícím kroku bude přistoupeno k implementaci GUI aplikace. Jako předloha k vytvoření GUI bude použit již vytvořený návrh. V návrhu jsou již přesně určeny prvky pro vytvoření GUI, jejich rozmístění a jsou také nadefinovány přechody mezi obrazovkami aplikace. Po implementaci GUI bude přistoupeno k jeho propojení s jednotlivými oblastmi funkční vrstvy. Následně bude přistoupeno k implementaci gest usnadňujících práci s aplikací.

V posledním sedmém kroku bude přistoupeno k testování aplikace. Testování aplikace bude probíhat emulací ve vývojovém prostředí a testováním přímo na zařízení s Windows Phone 8. Testování pomocí emulace bude probíhat průběžně při vývoji aplikace. Testování přímo na zařízení bude uskutečněno po vyvinutí celé aplikace. Následně z těchto testů budou vyvozeny závěry a případná vylepšení.

5 Požadavky na aplikaci

Aby bylo možné objektivně rozebrat požadavky na aplikaci, je nutné je rozdělit do dvou oblastí. A to na požadavky od zákazníka a na omezení (technologické požadavky) vyplývající z multiplatformního vývoje. Požadavky jsou uvedeny v kapitolách 5.1, respektive 5.2. Požadavky na aplikaci se obecně z hlediska softwarového inženýrství rozdělují na funkční a nefunkční požadavky. Z toho důvodu budou požadavky stanovené zadavatelem rozebrány z obou hledisek.

5.1 Požadavky stanovené zákazníkem

5.1.1 Funkční požadavky

Požadavky na funkcionalitu aplikace vyplývají z požadavků zadavatelské firmy, a jak je již psáno výše, představují funkční požadavky. Tím nejzákladnějším požadavkem je umožnit vhodné zobrazení mapových elementů. Dalším důležitým požadavkem je, aby aplikaci mohlo využívat více lidí. Z toho důvodu se musí uživatelé do aplikace přihlašovat. Není ale žádoucí, aby se uživatel přihlašoval při každém spuštění aplikace. Dále je požadováno, aby aplikaci na stejném zařízení mohlo využívat více lidí, proto musí aplikace umožňovat změnu přihlášené osoby.

Protože se jedná o aplikaci, která patří do kategorie GIS aplikací, musí tudíž poskytovat kompletní funkcionalitu této kategorie. Dalším požadavkem zadavatele je, aby vrstvy byly sdružovány do projektů. Aplikace tedy musí poskytovat možnost vytvořit nový projekt. Projekt může být vytvořený dvěma způsoby. Buď duplikací stávajícího projektu, nebo vytvoření jako zcela nový projekt. Dále aplikace musí poskytovat funkce pro práci s projekty. Tedy možnost projekt editovat nebo jej smazat. Dále musí zobrazovat projekty dostupné pro konkrétního uživatele. Aplikace také musí obsahovat možnost projekt uzamknout pro práci.

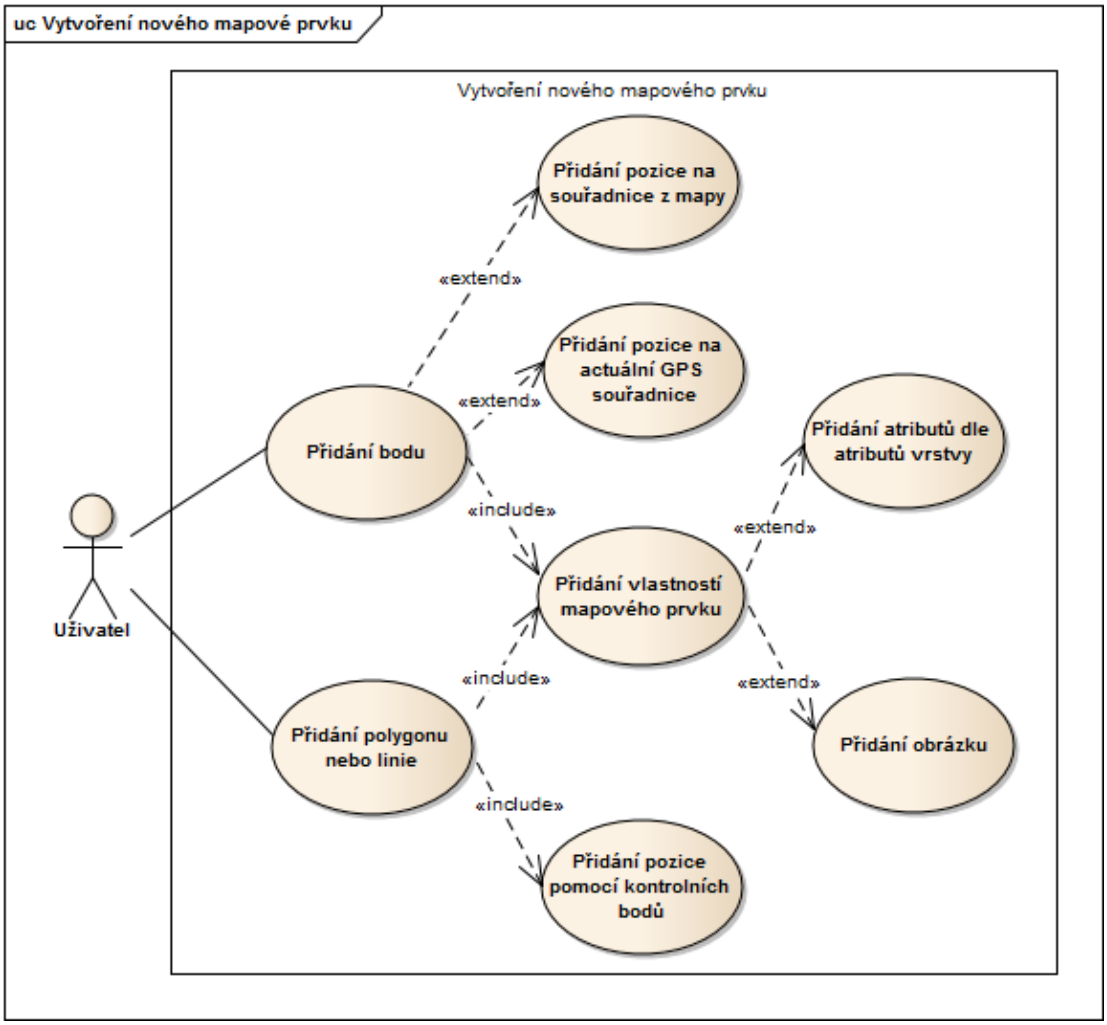
Aplikace musí být schopna uživateli zobrazit dostupné vrstvy v daném projektu. Dále musí poskytovat funkcionalitu pro sdružování objektů do konkrétní vrstvy. Do určitého projektu tedy musí být uživatel schopen přidávat nové vrstvy. Uživatel musí mít možnost upravovat vlastnosti konkrétní vrstvy nebo vrstvu odebrat. Dále musí být uživateli poskytnuta možnost výběru vrstev, které budou v daném projektu zobrazovány. Do konkrétní vrstvy musí být uživatel schopen přidávat různé objekty. Objekty mohou být tří typů:

- bod
- linie
- polygon

Dále musí být umožněno editovat vlastnosti konkrétního objektu nebo jej odebrat. Uživatel by měl také být schopen k objektu připojit doplňkové informace, např. obrázek.

V neposlední řadě musí mít uživatel možnost si změnit podkladovou mapu. Protože aplikaci může využívat jedna osoba na více zařízeních, musí aplikace data kvůli jejich aktuálnosti synchronizovat.

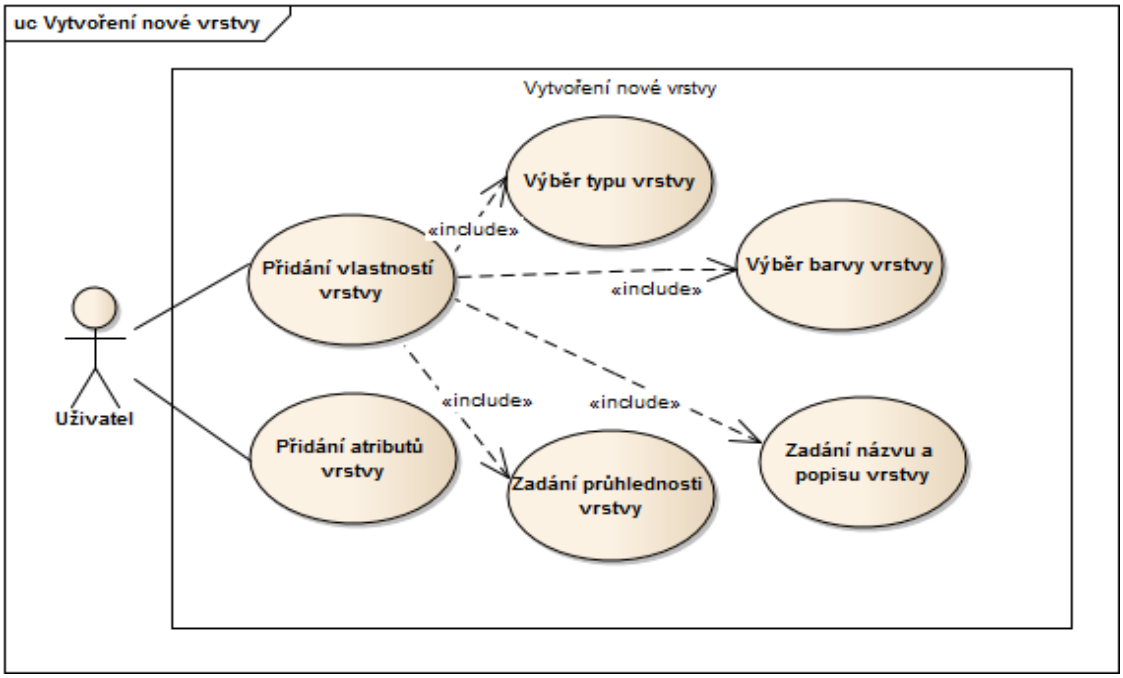
Funkční požadavky stanovené zákazníkem jsou znázorněny v příloze B pomocí diagramu případu užití. Pro lepší pochopení toho, jaké činnosti může uživatel v rámci určité funkcionality vykonávat, je nutné případy užití konkretizovat. Jako příklady konkretizace případu užití byly vybrány části vytvoření nové vrstvy a vytvoření mapového prvku.



Obr. 9 Případ užití – vytvoření mapového prvku

Konkretizace případu užití pro vytvoření mapového prvku je znázorněn na Obr. 9. Jedinou rolí v diagramu je role „*Uživatel*“. Ten může při přidání mapového prvku přidat vlastnosti mapového prvku a přidat pozici, kam bude mapový prvek umístěn. Přidání pozice mapového prvku se liší podle typu vrstvy. Pokud uživatel chce do bodové vrstvy přidat bod, může jeho pozici určit pomocí souřadnic z mapy nebo podle GPS souřadnic aktuální pozice uživatele. Protože se při přidání pozice

bodu má uživatel na výběr ze dvou možností, je použita vazba `extend`. Pokud chce uživatel do polygonové/liniové vrstvy přidat polygon/linii, tak pozici určuje pomocí kontrolních bodů. Dále vždy uživatel přidává vlastnosti mapového prvku. To znamená, že přidá atributy dle atributů nadefinovaných ve vrstvě a dále přidá obrázek.



Obr. 10 Případ užití – vytvoření nové mapové vrstvy

Konkretizace případu užití pro vytvoření nové vrstvy je znázorněna na Obr. 10. Jedinou rolí v diagramu je role „*Uživatel*“. Ten může při vytváření vrstvy přidat její vlastnosti nebo atributy. Přidání vlastností vrstvy v sobě vždy zahrnuje výběr typu vrstvy, barvy, kterou se budou její mapové prvky vykreslovat, zadání názvu a popisu a zadání průhlednosti vrstvy. Protože se tyto činnosti vždy vykonávají v rámci přidání vlastností vrstvy, je pro propojení činností použita vazba `include`.

5.1.2 Nefunkční požadavky

Nejzásadnějším požadavkem v této kapitole je platforma, pro kterou bude aplikace vyvíjena. Aplikace tedy bude vyvíjena pro platformu MS Windows Phone, konkrétně ve verzi 8. Dalším požadavkem je, aby databáze pro uložení dat byla umístěna na severu. Aplikace by také neměla být závislá na použitém hardware. Protože s aplikací by mělo být možné pracovat ve skutečném čase, nesmí doba odezvy serveru nebo databáze trvat déle než 10 sekund. Aplikace dále musí splňovat bezpečnostní požadavky. Těmi jsou především autorizace a autentizace uživatele a integritní omezení na přenášené zprávy tak, aby nebylo možné zprávy měnit.

5.2 Omezení vyplývající z multiplatformního vývoje

Jednotlivé požadavky byly vytvořeny vývojovým týmem pracujícím na projektu zastřešujícím vývoj aplikace pro platformy mobilních operačních systémů Android, iOS a Windows Phone. Typ těchto požadavků vychází z myšlenky jednotného vzhledu a funkcionality napříč všemi výše zmíněnými platformami. Jedná se o tyto požadavky:

- Vzhled GUI
- Způsob komunikace se serverem a databází
- Synchronizace dat a řešení konfliktních situací

5.2.1 Vzhled GUI

Jak již bylo popsáno výše, požadavky na vzhled GUI vycházejí z myšlenky stejného vzhledu napříč platformami mobilních operačních systémů Android, iOS a Windows phone. Jako předloha pro naplnění toho požadavku byl použit návrh GUI pro platformu iOS.

Protože vytvoření GUI na mobilních platformách probíhá propojováním jednolitých obrazovek (ve Windows phone „*pages*“) je návrh GUI tvořen sadou obrazovek. Návrh GUI také obsahuje přesně definované přechody mezi jednotlivými obrazovkami. Díky tomu jsou také přesně dány akce, které se mají vykonat při práci s určitým prvkem GUI. Návrh obrazovek s nadefinovanými přechody a akcemi je znázorněn v příloze A. Pomocí návrhu jsou také přesně dané prvky pro vytvoření GUI. Je také určené přesné rozmístění prvku na obrazovce.

Například z obrázku v příloze A je patrné, že nejdůležitějším prvkem první obrazovky je mapa pro zobrazování uložených dat. V levém horním rohu je umístěno tlačítko pro přechod do nastavení aplikace. V pravém horním rohu jsou umístěna tlačítka pro uzamčení a zoom mapy. Po stisku tlačítka pro nastavení je zobrazena obrazovka s nastavením aplikace, kde si uživatel například může ze seznamu aktivních vrstev vybírat vrstvy pro zobrazení. Po přechodu na další obrazovku si může zvolit typ podkladové mapy.

V návrhu jsou také určeny gesta, které bude aplikace podporovat. Je také nadefinováno, které gesto je použito nad určitou obrazovkou. Aplikace musí podporovat gesta:

- Swipe – pro přepínání obrazovek při práci s projekty a při práci s určitým objektem (mapový prvek)
- Pinch and zoom – pro zoom nad mapou

5.2.2 Způsob komunikace se serverem a databází

Databáze aplikace je umístěna na lokálním serveru. Do databáze se přistupuje pomocí rozhraní REST. Rozhraní REST k databázi přistupuje pomocí metod pro čtení, zápis, modifikaci a odebrání dat, tedy pomocí metod `READ`, `CREATE`, `UPDATE`, `DELETE`. Aby bylo možné se serverem komunikovat, je nutné zvolit

komunikační protokol. Z principu přístupu k serveru pomocí rozhraní REST vyplývá, že je výhodné, aby aplikace ke komunikaci se serverem využívala protokol HTTP. Při dotazu na server se vždy musí v hlavičce protokolu HTTP určit:

- Adresa pro dotaz, specifikující danou metodu v rozraní REST
- Akceptační typ pro odpověď serveru
- Údaje pro autentizaci uživatele
- Typ HTTP komunikace

Využití protokolu HTTP je logické už pouze z hlediska určení typu komunikace, kdy v hlavičce protokolu se vždy uvádí metoda `GET`, `PUT`, `POST`, `DELETE`. Vzájemná podoba metod je zobrazena v Tab. 2.

Tab. 2 Znázornění podobnosti metod rozhraní REST a protokolu HTTP

Rozhraní REST	Protokol HTTP
`Read`	`GET`
`Create`	`POST`
`Update`	`PUT`
`Delete`	`DELETE`

Předmět komunikace mezi aplikací a serverem je výměna dat. Takto vyměňovaná data jsou reprezentována JSON objekty. JSON objekt nabývá jednoho z devíti typů, které jsou níže detailněji popsány. Proto jako akceptační typ pro odpověď serveru se v HTTP hlavičce uvádí `application/json`. Každý JSON objekt je reprezentován aplikačními daty (data použitá jako vlastnosti objektů v objektově orientovaném návrhu) a meta daty. Meta data vypovídají o režijních vlastnostech dat (stáří dat a výsledky synchronizace). Požadavky na aplikační data jednotlivých JSON objektů jsou uvedeny v tabulkách 3 až 8. Jak již bylo zmíněno výše, díky použití REST rozhraní je každý dotaz obsahující JSON objekt posílán na specifickou adresu. Takto zaslaný dotaz na specifickou adresu s definovanou HTTP metodou přesně určuje akce, které se mají na serveru vykonat.

Objekt „*Project*“

Tab. 3 Aplikační data JSON objektu Project

Aplikační data	Popis
Name	Jednoznačný identifikátor projekt
Scheme	Identifikátor vlastníka projektu
Title	Název projektu
Description	Popis projektu

Prostřednictvím toho JSON objektu se vyměňují data o projektech. V Tab. 3 jsou popsány aplikační data objektu s jejich významem.

Objekt „*Layer*“

Tab. 4 Aplikační data JSON objektu Layer

Aplikační data	Popis
Name	Jednoznačný identifikátor vrstvy
Scheme	Identifikátor vlastníka vrstvy
Title	Název vrstvy
Type	Typ vrstvy (Point, Line, Polygone)
Description	Popis vrstvy

Tento JSON objekt slouží k výměně dat o vrstvách. V Tab. 4 jsou popsány aplikační data objektu s jejich významem. Na základě typu vrstvy se následně určuje, jaký typ objektu „feature“ se bude používat.

Objekt „*Project-Layer*“

Tab. 5 Aplikační data JSON objektu Project-Layer

Aplikační data	Popis
Color	Barva pro vykreslení vrstvy
Visibility	Viditelnost vrstvy
Transparency	Průhlednost
Zoom	Přiblížení

Tento JSON objekt slouží k výměně podrobnějších dat o vrstvách. V Tab. 5 jsou popsána aplikační data objektu s jejich významem. Jednotlivé vrstvy mohou být obsaženy ve více projektech, proto tento JSON propojuje projekt s konkrétní vrstvou.

Objekt „*Layer-Attribute*“

Tab. 6 Aplikační data JSON objektu Layer-Attribute

Aplikační data	Popis
Name	Název atributu
Type	Určení typu atributu

Tento JSON objekt slouží k výměně dat specifikující atributy vrstvy. V Tab. 6 jsou popsány aplikační data objektu s jejich významem. Data pouze specifikují, jaké vlastnosti atribut bude mít. Data neobsahují skutečnou hodnotu atributu. Ta je zasílána jiným způsobem, který je popsán níže.

Objekt „*Feature-(Point, Line, Poylygone)*“

Tab. 7 Aplikační data JSON objektu Feature-(Point, Line, Polygone)

Aplikační data	Popis
GeomId	Identifikační číslo mapového prvku
Created	Datum vytvoření
Last_edit	Datum poslední editace
Latitude	Zeměpisná délka a šířka
Longitude	
Height	Nadmořská výška
Precision	Přesnost naměřených dat
Bbox	Pouze u typu line a polygone

Pomocí toho JSON objektu se vyměňují data specifikující údaje o prvku určité vrstvy. Může se jednat například o jeden bod nebo linii. V Tab. 7 jsou popsány aplikační data objektu s jejich významem. Data především reprezentují údaje o poloze daného prvku.

Objekt „*Feature-Attribute*“

Tab. 8 Aplikační data JSON objektu Feature-Attriute

Aplikační data	Popis
Name	Název atributu
Value	Hodnota

Pomocí toho JSON objektu se vyměňují data vypovídající o obsahu atributů určitého prvku. V Tab. 8 jsou popsány aplikační data objektu s jejich významem. Vlastnosti atributu jsou přejaty z objektu `Layer-Attribute`.
Takto určené požadavky na jednotlivé JSON objekty byly použity pro vytvoření kostry nejnižší vrstvy aplikace.

5.2.3 Synchronizace dat a řešení konfliktních situací

Synchronizace dat je důležitá z toho důvodu, že každý uživatel může aplikaci využívat současně na různých zařízeních. Z toho vyplývají požadavky na informace, které každý objekt musí obsahovat. Každý objekt ve svých datech musí uchovávat informace zobrazené v Tab. 9.

Tab. 9 Meta data objektu

Meta data	Popis
Sync_state	Stav synchronizovaného objektu
Sync_result	Výsledek synchronizace
hash	Časová známka poslední synchronizace

Vlastnost `Sync_state` slouží k určení stavu objektu, například zda byl objekt už někdy synchronizován. Jednotlivé stavy, kterých objekt může nabývat, jsou obsaženy v Tab. 10. `Sync_result` vypovídá o tom, jak celá synchronizace skončila.

Vlastnost `hash` osahuj přesné určení času poslední synchronizace. Obsahuje tedy jak informace o datu a tak i o čase. `Hash` se používá i při výměně dat mezi serverem a aplikací. Důležitý je zejména při odebírání objektu z databáze. Časovou známku spolu s ostatními daty obdrží objekt jako odpověď serveru na určitý dotaz.

Tab. 10 Synchronizační stavy jednotlivých objektů

Sync_state	Popis
NEW	Nový objekt, ale není synchronizovaný se serverem
POSTING	Nový objekt již synchronizovaný
UPDATE	Upravený objekt, ale není synchronizovaný se serverem
PUTING	Upravený, synchronizovaný objekt
DELETE	Smazaný objekt v zařízení, ale není smazaný ze serveru
DELETING	Objekt smazaný jak ze zařízení, tak ze serveru

Funkčnost synchronizace vychází z inspirace protokolu GCM (Google Cloud-Message). Celá synchronizace musí probíhat následujícím způsobem. Jakmile se začne pracovat s určitým projektem, tak se ze serveru synchronně stáhne projekt a vrstvy patřící k tomuto projektu. Dokud probíhá stahování, nemůže uživatel s aplikací pracovat. Jakmile bude vše staženo, tak se zařízení na serveru zaregistruje jako posluchač a je mu přiřazeno unikátní id. Poté, pokud je potřeba synchronizovat, tak zařízení pod unikátním id zašle na server dotaz se seznam vrstev, které aktuálně využívá. Server zašle zařízení odpověď, která bude osahovat seznam vrstev. Pro každou vrstvu bude odpověď obsahovat seznam modifikací v datech. Následně si zařízení aktualizuje svá data. Jakmile zařízení ukončí práci s projektem, odregistruje ze serveru

6 Konkretizace cíle práce

V kapitole 2.3 jsou uvedeny orientační cíle, kterých má být dosaženo. Na základě dosavadní analýzy je možné odpovědět na následující otázky. Tyto odpovědi zcela konkrétně určují problémy, které musí být v této práci vyřešeny. Jedná se o následující otázky:

- Jak reprezentovat prvky vrstev?
- Jaká funkcionalita bude poskytována pro práci s mapovými prvky?
- Jaké požadavky zákazníka mají být splněny?

Jako nástroj pro zobrazování mapových prvků (bod, linie, polygon) bude zvolena mapa. Jednotlivé mapové prvky budou zobrazovány na základě uživatelova nastavení vrstev. To znamená, že z vlastností vrstev budou mapové prvky přebírat barvu, průhlednost, přiblížení a viditelnost. Bod bude vykreslován jako čtverec o konstantní velikosti, linie jako křivka se zvýrazněnými body a polygon jako mnohoúhelník opět se zvýrazněnými body.

Další nevyjasněnou otázkou je poskytovaná funkcionalita pro práci s mapovými prvky. Uživatel bude moci mapové prvky přidávat nebo editovat. Dále bude moci přidávat, editovat nebo odebírat vlastnosti jednotlivých mapových prvků. To, jakým způsobem budou mapové prvky přidávány, závisí na jejich typu. Přidávání bude probíhat ve dvou krocích. V prvním kroku se určí poloha mapového prvku. Pokud se bude jednat o typ bod, tak uživatel určí polohu kliknutím do mapy nebo pomocí speciálního tlačítka na svoji aktuální pozici. Pokud se bude jednat o linii nebo polygon, uživatel bude postupným klikáním do mapy přidávat kontrolní body vymezující polohu budoucího mapového prvku. Při přidávání kontrolních bodů se bude uživatel moci vracet pomocí odebírání posledních přidaných bodů. V druhém kroku bude uživatel vyzván k zadání případných vlastností mapového prvku. Editace mapového prvku bude opět probíhat ve dvou oddělených krocích. Uživatel bude moci upravovat pozici mapového prvku pomocí kliku na bod (v případě linie nebo polygonu kontrolní bod), čímž daný bod zaktivuje, následně bod přesune na jinou pozici na mapě a úpravu potvrdí klikem. Dále uživatel bude moci pomocí určených obrazovek editovat nebo odebírat vlastnosti mapového prvku.

Posledním nevyjasněnou otázkou je, jaké požadavky zákazníka mají být splněny. Jedná se především o požadavky na funkcionalitu a technické vlastnosti aplikace. Tyto požadavky jsou detailněji rozepsány v kapitole 5.1.

7 Návrh a implementace aplikace

Před popisem samotného návrhu a implementace vyvíjené GIS aplikace, je nejdříve nutné si připomenout hlavní cíle aplikace, jakou má aplikace poskytovat funkcionalitu a jak by měla aplikace pracovat.

Jak je uvedeno v kapitole 2.3, cílem práce je vytvoření GIS aplikace pro mobilní platformu Windows Phone 8, která bude poskytovat komplexní funkcionalitu danou typem aplikace a funkcionalitu požadovanou zadavatelem. Pro lepší pochopení funkcionality, kterou aplikace poskytuje, byl vytvořen diagram případů užití zobrazený v příloze B.

Diagram obsahuje jedinou roli s názvem „uživatel“ a jednotlivé případy užití. Tyto případy užití reprezentují body požadované funkcionality. Mezi případy užití jsou také vytvořeny závislosti. Pokud je použita závislost `extend`, znamená to podmíněnou závislost mezi případy užití. Například tato závislost je použita při vytvoření vrstvy, kdy uživatel může přidat vrstvu bez atributů. Pokud je použita závislost `includce`, znamená to přímou závislost mezi případy užití. Například, pokud je smazán projekt, jsou automaticky smazány všechny vrstvy v projektu a další prvky vrstev. Aplikace tedy umožňuje:

- Pracovat s projekty pomocí:
    - Vytvoření nového projektu
    - Vytvoření kopie stávajícího projektu
    - Editace nebo smazání projektu
- Pracovat s vrstvami pomocí:
    - Vytvoření nové vrstvy
    - Přidání vrstvy z jiného projektu
    - Editace nebo smazání vrstvy
    - Exportu vrstvy
    - Změny pořadí vrstev
    - Změny viditelnosti vrstev
    - Přidání, editace nebo smazání atributů vrstvy
- Pracovat s mapovými prvky určité vrstvy pomocí:
    - Vytvoření nového mapového prvku
    - Editace nebo smazání mapového prvku
    - Přidání, editace nebo smazání atributů mapového prvku
- Pracovat s mapou pro vykreslení dat pomocí:
    - Změny podkladové mapy
    - Změny přiblížení mapy

Pro vývoj aplikace byl zvolen programovací jazyk C#, který je součástí frameworku .NET. Jako verze frameworku .NET byla zvolena verze 4.5. Programovací jazyk C# a to především z důvodu zachování kompatibility s verzí Windows Phone 7.x. Dále z toho důvodu, že oproti jazykům C++ a Visual Basic .NET, pomocí kterých lze také vyvíjet aplikace pro platformu Windows Phone 8, jak je uvedeno

v kapitole 3.2.1, je lépe rozšiřitelný o externí knihovny. K implementaci zdrojového kódu bylo použito vývojové prostředí Microsoft Visual Studio ve verzi 12. Aby bylo možné vyvíjet aplikaci, bylo do vývojového prostředí doinstalováno rozšíření Windows Phone SDK ve verzi 8.0. Funkce a prostředky použitelné k vývoji aplikace, které doinstalování rozšíření Windows Phone SDK přineslo, jsou popsány v kapitole 3.2.2.

Do vývojového prostředí byly dále doinstalovány externí knihovna NewtonSoft pro práci s JSON soubory, která je popsána v kapitole 3.3.2 a balík Windows Phone Toolkit od organizace CodePlex, který obsahuje nestandardní prvky pro tvorbu GUI. Doinstalování tohoto balíku umožnilo využívat prvky jako:

- `DateTimePickers`
- `ListPicker`
- `ColorPicker`
- `ToggleSwitch`
- `CustomeMessageBox`

(The Windows Phone Toolkit, 15.8.2013)

7.1 Návrh a implementace databázové vrstvy

Návrh lokální databáze je vytvořen pomocí ERD (Entitně-relační model). Tento model vychází z omezení uvedených v kapitole 5.2.2 Způsob komunikace se serverem a databází. ERD lokální databáze je zobrazen v příloze C. Návrh databáze je upraven podle požadavků stanovených implementací objektového modelu, například vytvoření tabulky `Points`. Pomocí ERD jsou tedy postupně navrženy tabulky `Projects`, `Project-Layers`, `Layers`, `Layer-Attbributes`, `Features`, `Feature-Attbriutes`, `Points` a vazby mezi jednotlivými tabulkami. Podle předlohy návrhu databáze byla pomocí nástroje SQL CE (Microft SQL Server Compact) vytvořena lokální databáze, která byla uložena do souboru s příponou `SDF`. Poté bylo přikročeno k samotné implementaci lokální databáze pomocí technologie LINQ to SQL. Tato technologie je detailněji popsána v kapitole 3.3.1. Lokální databáze byla vytvořena níže uvedeným způsobem.

Pro každou tabulku z ERD je vytvořena datová třída, např. datová třída `ProjectLayerDB`. Tabulky databáze byly následně mapovány vytvořené na datové třídy. Aby bylo možné používat databázi, byl vytvořen datový kontext reprezentovaný třídou `DBContext`. Do datového kontextu byly vytvořeny metody pro práci s databází. Třídy a jejich vazby jsou zobrazeny v příloze D.

Třída DBContext

Tab. 11 Třída DBContext

Název metody / atributu	Návratový / datový typ	Popis
db	`DataContext`	Reprezentace databáze
Projects	`Table<ProjectDB>`	Tabulka reprezentující projekty
ProjectsLayers	`Table<ProjectsLayerDB>`	Tabulka reprezentující specifikaci vrstev
Layers	`Table<LayerDB>`	Tabulka reprezentující vrstvy
Layer-Attributes	`Table<LayerAttributeDB>`	Tabulka reprezentující atributy vrstev
Features	`Table<FeatureDB>`	Tabulka reprezentující mapové prvky
Feature-Attributes	`Table<FeatureAttributeDB>`	Tabulka reprezentující atributy mapových prvků
Points	`Table<PointDB>`	Tabulka reprezentující souřadnice mapových prvků
Metody pro přidání řádku do tabulky		
Metody pro odebrání řádků z tabulky		
Metody pro editaci záznamu v tabulce		
Metody pro získání seznamu všech záznamů v tabulce		
Metody pro získání právě jednoho záznamu v tabulce		

Jak již bylo uvedeno výše, třída `DBContext` reprezentuje datový kontext pro práci s databází. Datový kontext znamená, že třída zastupuje databázi. Proto klíčovou vlastností této třídy je vlastnost `db`. Třída dále obsahuje jednotlivé tabulky datových tříd.

V konstruktoru třídy probíhá inicializace databáze. Nejprve je vytvořena vlastnost `db`, jako instance třídy `DataContext`, kdy v parametru je předán připojovací řetězec k databázi. Připojovací řetězec má následující tvar: „*appdata:CMlocaldatabase.sdf*“, což znamená, že databáze je umístěna v instalační složce aplikace. Dále jsou z inicializovány jednotlivé tabulky. Inicializace je provedena pomocí metody `GetTable()` vlastnosti `db`. Pokud není v instalační složce nalezen soubor s databází, je automaticky vytvořen spolu s celou databází.

Veškeré metody této třídy jsou statické a to z důvodu použitelnosti databáze v ostatních třídách funkční vrstvy, protože datový kontext musí být v aplikaci vytvořený pouze jednou. Velmi zajímavými metodami z pohledu implementace jsou

metody pro úpravu záznamu v databázi, jako je například metoda `updateProjectLayer()`. Jako parametr se této třídě předává instance datové třídy `ProjectLayerDB`. V metodě se nejprve ve vlastnosti (tabulce) `ProjectLayers` vyhledá podle dotazu pro výběr (`select`) záznam pro úpravu. Vyhledání záznamu pro úpravu probíhání podle vlastnosti `id` parametru metody. Následně se ve vyhledaném záznamu upraví veškeré vlastnosti a celý proces se potvrdí metodou `SubmitChanges()` vlastnosti `db`.

Třída ProjectLayerDB

Tab. 12 Datová třída ProjectLayerDB

Název metody / atributu	Návratový / datový typ	Popis
ID	`int`	Id objektu
Color	`string`	Barva vrstvy
Zoom	`double`	Zoom vrstvy
Visibility	`byte`	Viditelnost vrstvy
Transparency	`double`	Průhlednost vrstvy
LayerID	`int`	Id vrstvy
ProjectID	`int`	Id projektu
Project	`EntityRef<ProjectDB>`	Projekt
Layer	`EntityRef<LayerDB>`	Vrstva

Jak bylo uvedeno výše, tato třída slouží k mapování tabulky, konkrétně tabulky `ProjectLayers`. Jedná se o tabulku vzniklou rozpadem vazby M:N mezi tabulkami `Projects` a `Layers`. Kód pro reprezentaci této třídy je uveden v příloze E. Při definici třídy se uvádí mapovací atribut `Table` s vlastností `Name`, kterou nastaví na hodnotu `Project-Layers`.

Jednotlivé atributy jsou zpřístupněny přes accesory. Na tyto accesory jsou použity mapovací atributy `Column` a `Association`. Mapovacímu atributu `Column` s vlastností `Name` se nastavuje název sloupce pro mapování. U atributu id se ještě uvádí vlastnost mapovacího atributu `IsPrimaryKey=true`. Při vytvoření vazeb mezi tabulkami, pro které se používají atributy `Project` a `Layer` se využije mapovací atribut `Association`, kterému se vlastnost `ThisKey` nastaví na hodnotu `LayerID` resp. `ProjectID`, vlastnost `OtherKey` na hodnotu `LayerID` resp. `ProjectID` a vlastnost `storage` na hodnotu `Layer` resp. `Project`. Pro vytvoření vazby se použije kolekce `EntityRef`, které se jako generický typ nastaví datová třída, se kterou vytváříme vazbu, tedy typ `LayerDB` resp. `ProjectDB`.

7.2 Návrh a implementace funkční vrstvy

Objektový model funkční vrstvy je vytvořen na základě Analytického modelu tříd uvedeném v příloze D. Analytický model je vytvořen podle předlohy JSON objektů uvedených v kapitole 5.2.2, který je dále doplněn o omezení vzniklé implementací lokální databáze, uvedených výše v kapitole 7.1.

Analytický model je rozdělen do několika modulů. Jedná se o moduly týkající se lokální databáze, objektů pro reprezentaci dat a práce s HTTP komunikací. Vybrané části týkající se sekce o lokální databázi jsou popsány výše v kapitole 7.1. Jedná se především o třídu `DBContext` a příklad datové třídy `ProjectLayerDB`. Jak je patrné z analytického modelu, třída `DBContext` vzniká kompozicí tabulek jednotlivých datových tříd. Datové třídy jsou dále s třídou `DBContext` asociovány a to z důvodu práce s tabulkami databáze.

Dalším modulem je modul pro reprezentaci dat v aplikaci. Jedná se o třídy vzniklé dle předlohy JSON objektů. Jsou to tedy třídy `Projects`, `ProjectLayer`, `Layer`, `Layer-Attbribute`, `Feature`, `FeatureAttbriute`, `Point`. Tyto třídy obsahují stejné vlastnosti jako JSON objekty, proto nebudou později detailněji rozebírány. Jak je patrné z analytického modelu, tyto objekty tvoří stromovou strukturu, proto kromě již uvedených vlastností obsahují kolekce dalších objektů. Třída `Projects` vzniká kolekcí objektů `ProjectLayer`, které dědí vlastnosti z objektu `Layer`. Třída `Layer` dále vzniká agregací objektů `Feature`, které vznikají kompozicí objektu `Point` a agregací objektu `FeatureAttribute`. Tyto třídy dále obsahují parametrické konstruktory pro nastavování vlastností. Dále obsahují metody pro práci s kolekcemi objektů dle výše uvedených vazeb. Tyto třídy dále splňují rozhraní `IParseComunication` pro manipulaci dat mezi serverem a daným objektem. Každá třída tedy obsahuje metody `read()`, `create()`, `update()` a `delete()`. Třídy `Project` a `Layer` dále osahují metody `readAll()` pro získání veškerých projektů nebo vrtem bez podřízených objektů.

Do posledního modulu spadá třída `SyncMaster`, která zajišťuje HTTP komunikaci se serverem. Třídy pro reprezentaci dat jsou s touto třídou asociovány a to z důvodu přesunu stažených dat ze serveru nebo dat k nahrání na server. Třída `SyncMaster` já níže detailněji popsána.

Třída SyncMaster

Tab. 13 Třída SyncMaster

Název metody / atributu	Návratový / datový typ	Popis
serverAddressPart1	`string`	První část adresy
httpClient	`HttpClient`	Klient pro přístup na server
read	`Task<String>`	Načtení dat ze serveru
post	`void`	Vytvoření záznamu na serveru
put	`void`	Úprava dat na serveru
delete	`void`	Smazání dat ze serveru

Jak je popsáno výše, třída slouží k výměně dat se server. Vlastnost `serverAddressPart1` je neměnná část URL adresy, čili je to pouze adresa serveru. Druhá vlastnost `httpClient` představuje klient pro komunikace se serverem. Umožňuje tedy používat metody pro přenos dat. Obě tyto vlastnosti jsou nastavovány v konstruktoru třídy. V konstruktoru je nejdříve vytvořena instance třídy `HttpClientHandler`, která umožňuje nastavit přihlašování údaje uživatele a které jsou použity při autorizaci uživatele. Následně je vytvořena instance třídy `HttpClient` kde jako parametr se předává vytvořený handler. Dále se v klientu nastavuje typ komunikace. Typ komunikace je nastavený jako „*application/json*“. Následně je v klientu nastaven základ URL adresy, který je také uložen do vlastnosti `serverAddressPart1`.

Pomocí výše uvedených metod `read()`, `post()`, `put()` a `delete()` aplikace komunikuje se serverem. Jako parametr se těmto metodám předává druhá část URL adresy specifikovaná adresami REST rozhraní. Než může být přikročeno ke komunikaci se serverem, musí se nejprve v každé metodě zkontrolovat správnost URL adresy. To je zajištěno kódem uvedeným v příloze F. Nejprve se otestuje syntaktická správnost adresy. Dále je ověřeno schéma adresy. Jako přípustná schémata pro URL adresu jsou schéma „*http*“ a „*https*“. Pokud tyto ověření proběhnou správě, je přikročeno ke komunikaci se server. Ta je zajištěna asynchronními metodami vlastnosti `httpClient`. Pokud chceme data ze serveru číst, jek tomu v metodě `read()` použita metoda `GetStringAsync()`. Pokud na serveru chceme vytvořit nový záznam, je k tomu v metodě `post()` použita metoda `postAsync()`. Pokud chceme upravit nějaký záznam, jek tomu v metodě `put()` použita metoda `putAsync()`. Pokud chceme na serveru smazat nějaký záznam, je k tomu v metodě `delete()` použita metoda `deleteAsync()`.

7.3 Chování vybraných části aplikace

K lepšímu pochopení jak probíhá komunikace mezi GUI a funkční, potažmo databázovou vrstvou jsou vytvořeny diagramy aktivit a sekvenční a stavové

diagramy. Tyto diagramy jsou vytvořeny pro vybrané části aplikace (vybranou funkcionalitu). Namodelovány tedy jsou diagramy pro funkcionalitu vytvoření nové vrstvy a vytvoření mapového prvku. Přehled ostatní funkcionality je zobrazen v příloze B.

7.3.1 Chování aplikace při vytvoření nové vrstvy

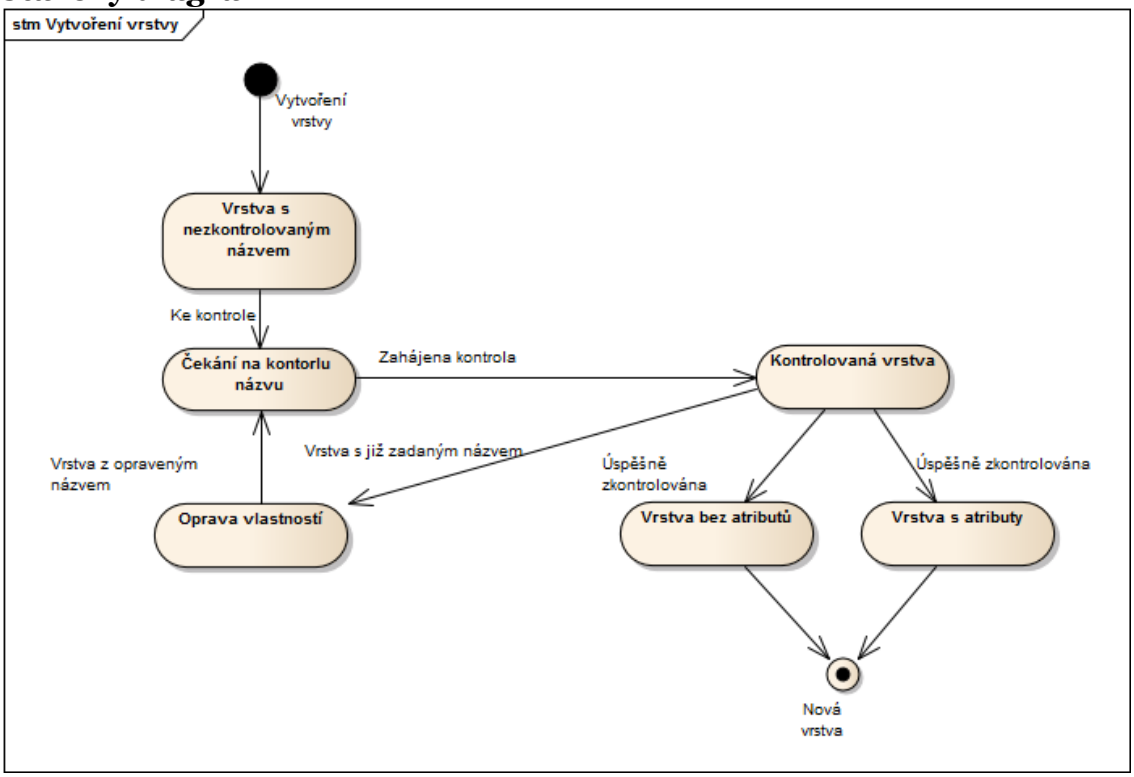
Sekvenční diagram

Sekvenční diagram znázorňující chování aplikace při vytváření nové vrstvy je uveden v příloze G. Jedinou rolí v diagramu je role „Uživatel“. Jednotlivé čáry života představují třídy aplikace. Třídy označené stereotypem `UI` představují třídy GUI a třídy označené stereotypem `base layer` představují třídy funkční vrstvy. V diagramu je dále znázorněna entita „*DB*“ představující lokální databázi.

Po vytvoření požadavku na zobrazení stránky `CreateLayerPage` se ihned zobrazí panel `AddNewLayerControl`, kde uživatel zadává vlastnosti vrstvy. Po vykonání `swipe` gesta (vazba se stereotypem `swipe`) na stránce `CreateLayerPage` je uživateli zobrazen panel `AddNewLayerControl` nebo panel `AddLAttributeControl`, kde uživatel může vytvářet atributy vrstvy. Pokud uživatel stiskne na stránce `CreateLayerPage` tlačítko „*Properties*“, je mu zobrazen panel `AddNewLayerControl`. Pokud uživatel stiskne na stránce `CreateLayerPage` tlačítko „*Attributes*“, je mu zobrazen panel `AddLAttributeControl`. Po stisku tlačítka na výběr barvy na panelu `AddNewLayerControl` je uživateli zobrazena stránka `ColorPickerPage` pro výběr barvy vrstvy.

Po stisku tlačítka „*Add*“ na stránce `CreateLayerPage` je pomocí metody `AddLayer()` třídy `Project` vytvořena instance třídy `ProjectLayer`, a protože tato třída je potomkem třídy `Layer`, je vytvořena i instance této třídy. Tímto je vytvořena vrstva. Dále jsou také vytvořeny případné atributy vrstvy jako instance třídy `LayerAttribute`. Následně jsou zaslány požadavky na vytvoření záznamu v lokální databázi. Pomocí metody `AddLayerAttribute()` třídy `DBContext` jvytvořen v databázi (vazba se stereotypem `insert`) v tabulce `LayerAttributes` záznam atributu. Dále pomocí metody `AddLayer()` třídy `DBContext` je vytvořen v databázi v tabulce `Layers` záznam vrstvy. Následně pomocí metody `AddProjectLayer()` třídy `DBContext` je v databázi v tabulce `ProjectLayers` vytvořen záznam projektové vrstvy.

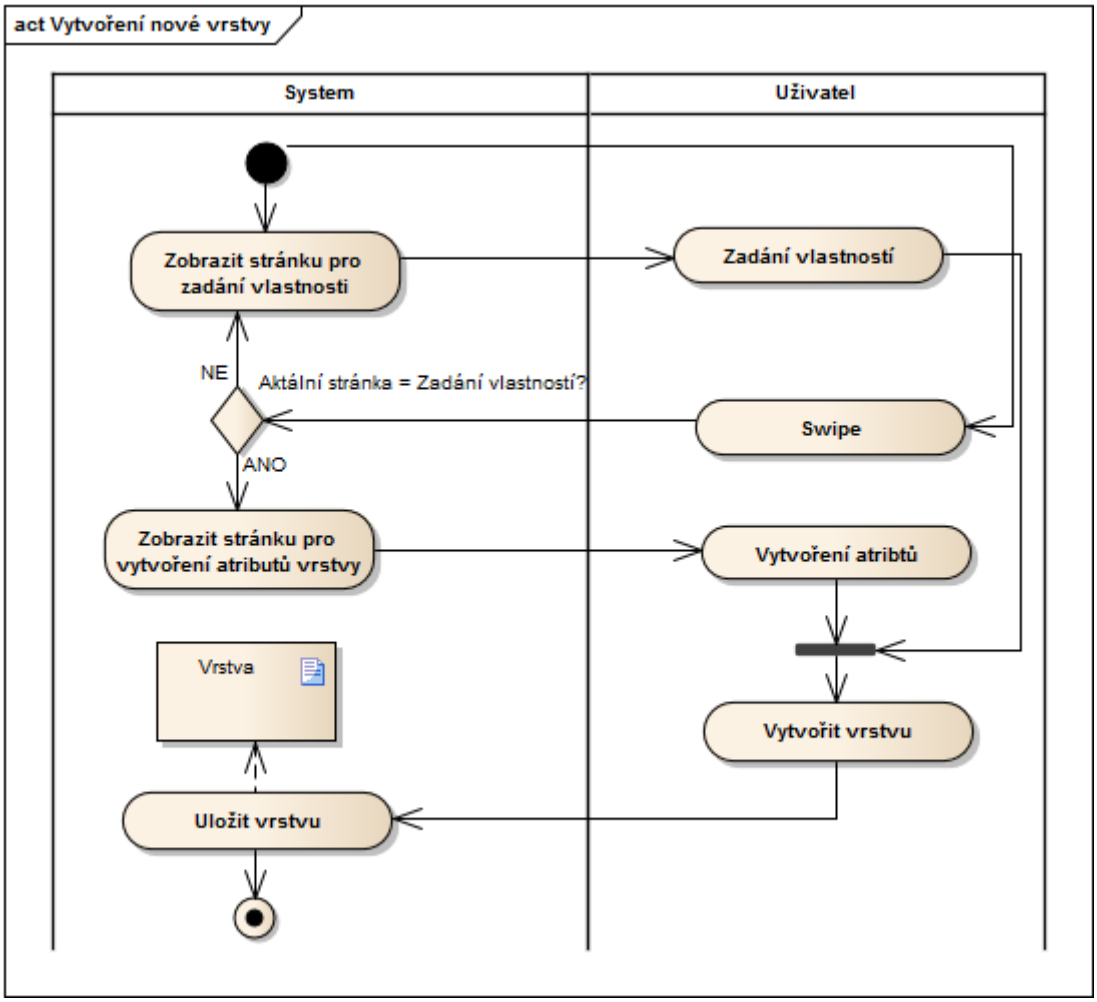
Stavový diagram



Obr. 11 Stavový diagram vytvářené vrstvy

Stavový diagram znázorňující stavy, kterými prochází vytvářená vrstva je znázorněn na Obr. 11. Z počátečního uzlu vychází vrstva pro stisku tlačítka pro přidání vrstvy. Takováto vrstva nemá doposud zkontrolovaný název. Následně se vrstva nachází ve stavu čekání na kontrolu názvu. Po zahájení kontroly názvu se vrstva nachází ve stavu kontroly vrstvy. Kontroluje se, zda se název vrstvy neshoduje s názvem již vytvořené vrstvy. Pokud se název shoduje, nachází se vrstva ve stavu opravy vlastností a následně opět ve stavu čekání na kontrolu s již opraveným názvem. Pokud je název správný, může být vytvářená vrstva ve stavu bez atributu nebo s atributy.

Diagram aktivit



Obr. 12 Diagram aktivit pro vytvoření nové vrstvy

Diagram aktivit znázorňující interakci uživatele se systém při vytváření nové vrstvy je zobrazen na Obr. 12. Diagram obsahuje dle rolí „Systém“ a „Uživatel“ dvě plavecké dráhy. Na začátku interakce systém zobrazí uživateli stránku pro zadání vlastností vrstvy. Dále uživatel může již vrstvu vytvořit nebo pomocí `swipe` gesta se přepnout na jinou stránku. Pokud je aktuálně zobrazena stránka pro zadání vlastností, je uživateli zobrazena stránka pro vytvoření atributů, kde uživatel následně vytvoří atributy. Následně pokud uživatel zvolí akci pro vytvoření vrstvy, tak systém vrstvu uloží a proces interakce končí.

7.3.2 Chování aplikace při vytvoření mapového prvku

Protože vytváření mapového prvku se liší podle typu vrstvy, do které se mapový prvek vytváří, jsou pro lepší přehlednost vytvořeny dva sekvenční diagramy.

Sekvenční diagram pro vytvoření mapového prvku – bod
Sekvenční diagram znázorňující chování aplikace při vytváření bodu je uveden v příloze H. Jedinou rolí v diagramu je role „Uživatel“. Jednotlivé čáry života představují třídy aplikace. Třídy označené stereotypem `UI` představují třídy GUI a třídy označené stereotypem `base layer` představují třídy funkční vrstvy. V diagramu je dále znázorněna entita „*DB*“ představující lokální databázi, entita „*Camera*“ představující fotoaparát a entita „*IsolatedStorage*“ představující paměť telefonu.

Po vytvoření požadavku na zobrazení stránky `MapAddPoint` je uživateli zobrazena stránka pro přidání bodu. Po kliknutí uživatele do mapy je vyvolána metoda `CMmap_Tap()`, která následně zobrazí stránku `AddPointPage` a na ní panel PointPiv1 pro zadání vlastností mapové prvku. Po stisku tlačítka „*Add current location*“ je vyvolána metoda `Add_Click()`, která následně také zobrazí stránku `AddPointPage` a na ní panel `PointPiv1` pro zadání vlastností mapového prvku.

Po zobrazení stránky `AddPointPage` může uživatel pomocí `swipe` gesta (vazba se stereotypem `swipe`) přepínat mezi panelem pro zadání vlastností (`PointPiv1`) nebo panelem pro přidání obrázku (`FeatureAttributesPiv3`). Přepnutí mezi panely může také vytvořit stiskem tlačítka „*Attributes*“, kdy je vyvolána metoda `Attributes_Click()` a zobrazen panel PointPiv1 nebo stiskem tlačítka „*Photo*“, kdy je vyvolána metoda `Photo_Click()` a zobrazen panel `FeatureAttributesPiv3`. Po stisku tlačítka „*Change Picture*“ je vyvolána metoda `Button_Click()` a následně je zobrazena systémová aplikace pro fotoaparát a po pomocí metody `cameraCaptureTask_Completed()` je vyfocený snímek načten do aplikace.

Po stisku tlačítka „*Save*“ na stránce `AddPointPage` je vyvolána `Save_Click()` pro uložení mapového prvku. Pomocí metody `SaveToMediaLibrary()` je nejprve uložen snímek to paměti telefonu. Dále pomocí metody `AddFeature()` třídy `Layer` je vytvořena instance třídy `Feature`. Pomocí metody `AddFeature()` třídy `DBContext` je v databázi v tabulce `Features` vytvořen záznam (vazba se stereotypem `insert`). Dále jsou vytvořeny instance třídy `FeatureAttribute`. Pomocí `metody AddFeatureAttribute()` třídy `DBContext` je v databázi v tabulce `FeatureAttributes` vytvořen záznam. Nakonec je vytvořena instance třídy Point a pomocí metody `AddPoint()` třídy `DBContext` je v databázi v tabulce `Points` vytvořen záznam. Po přidání mapového prvku jsou vrstvy pomocí metody `PaintLayers()` třídy `MapAddPoint` překresleny.

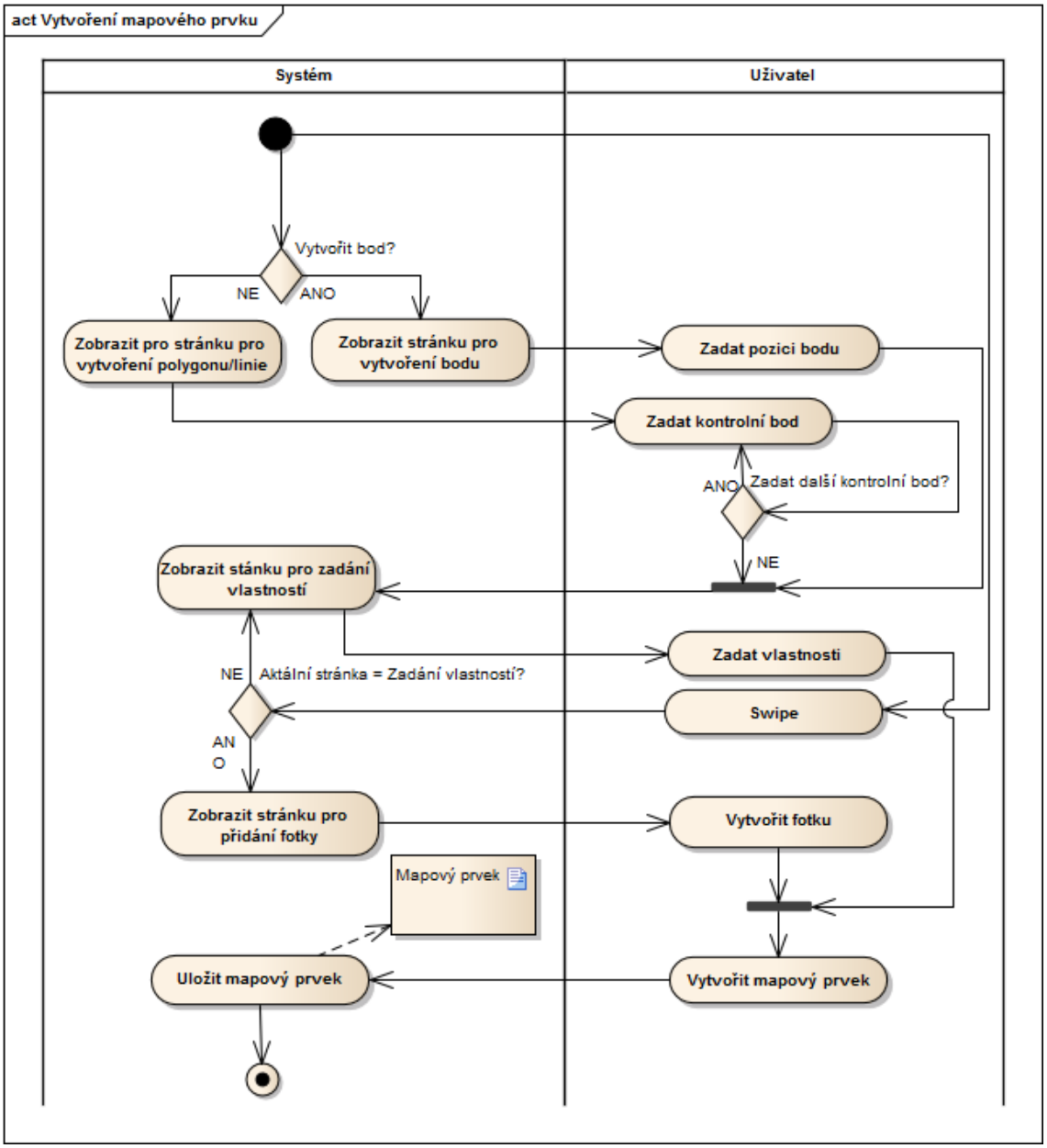
Sekvenční diagram pro vytvoření mapového prvku – polygon, linie
Tento diagram se oproti výše uvedenému sekvenčnímu diagramu liší pouze po stránce chování GUI. Z toho důvodu bude tento diagram popsán pouze se stránky GUI, popis interakce s funkční vrstvou je uveden výše v posledním odstavci. Diagram je uveden v příloze I.

Po vytvoření požadavku na zobrazení stránky `ManualPolygonPage` je uživateli zobrazena stránka pro přidání pozice mapového prvku. Pomocí kliknutí do mapy je vyvolána metoda `CMmap_Tap()`, která vykreslí pozici do mapy. Po stisku tlačítka „*Add point*“ je vyvolána metoda `Add_Click()`, pomocí které je přidán kontrolní bod a následně pomocí metody `PaintPoint()` jsou kontrolní body překresleny. Dále pokud uživatel přidává pozici polygonu, tak je pomocí metody `PaintNewPolygon()` téže třídy vykreslen polygon. Pokud uživatel přidává pozici linie, je pomocí metody `PaintNewLine()` téže třídy vykreslena linie. To samé se děje po stisku tlačítka „*Undo point*“, kdy je vyvolána metoda `Undo_Click()`. Po stisku tlačítka „*Finish*“ je vyvolána metoda `Finish_Click()` kdy je následně stránka `AddPolygonAttributePage` s panelem `AddPolPiv1`.

Po zobrazení stránky `AddPolygonAttributePage` může uživatel pomocí `swipe` gesta (vazba se stereotypem `swipe`) přepínat mezi panelem pro zadání vlastností (`AddPolPiv1`) nebo panelem pro přidání obrázku (`FeatureAttributesPiv3`). Po stisku tlačítka „*Attributes*“ je vyvolána metoda `Attributes_Click()` a následně je uživateli zobrazena stránka `AddPolPiv1`. Po stisknutí tlačítka „*Photo*“ je vyvolána metoda `Photo_Click()` pro zobrazení panelu `FeatureAttributesPiv3`. Stiskem tlačítka „*Change picture*“ je vyvolána metoda `Button_Click()` a následně je zobrazena systémová aplikace pro fotoaparát. Po zachycení snímku je snímek pomocí metody `cameraCaptureTask_Completed()` načten do aplikace.

Diagram aktivit

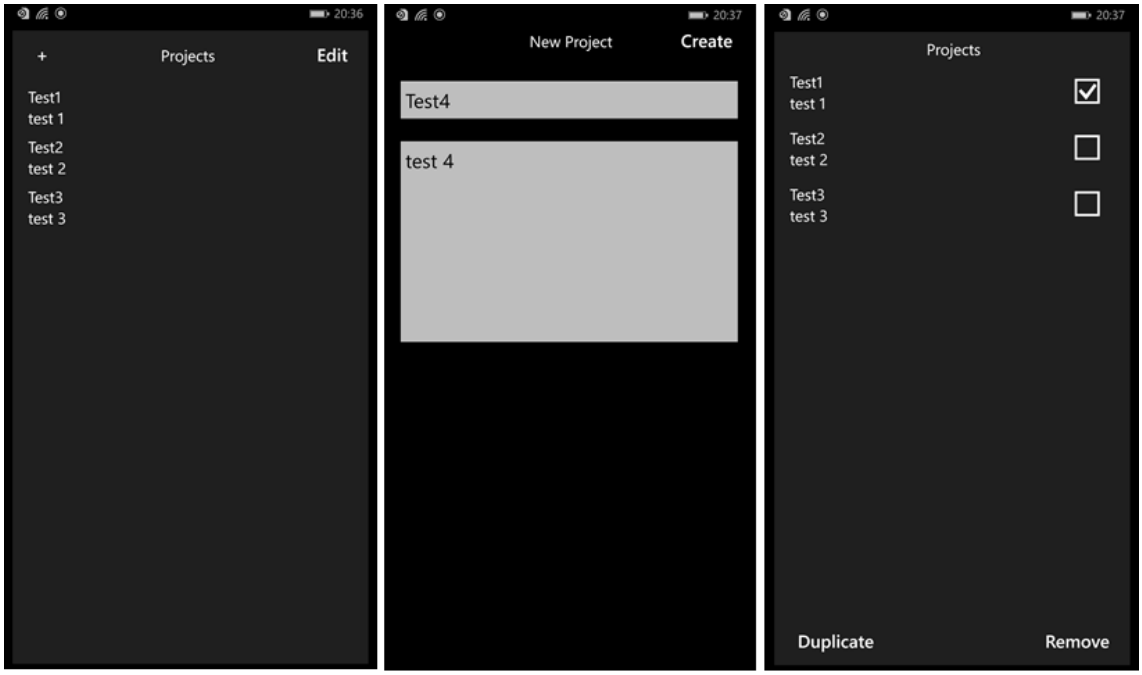
Diagram aktivit znázorňující interakci uživatele se systém při vytváření mapového prvku je zobrazen na Obr. 13. Diagram obsahuje dle rolí „Systém“ a „Uživatel“ dvě plavecké dráhy. Na začátku interakce systém dle typu vrstvy zobrazí uživateli stránku pro přidání pozice mapového prvku. Po přidání pozice mapového prvku je uživateli zobrazena stránka pro přidání vlastností. Dále uživatel může již mapový prvek vytvořit nebo pomocí `swipe` gesta se přepnout na jinou stránku. Pokud je aktuálně zobrazena stránka pro zadání vlastností, je uživateli zobrazena stránka pro vytvoření fotky, kterou následně uživatel pomocí fotoaparátu pořídí. Následně pokud uživatel zvolí akci pro vytvoření mapového prvku, tak jej systém uloží a proces interakce končí.



Obr. 13 Diagram aktivit přidání nového mapového prvku

7.4 Popis GUI

Tato kapitola se zabývá popisem uživatelského rozhraní aplikace. GUI je popisováno z uživatelského hlediska. Jsou tedy detailněji popisovány akce, které uživatel může na dané obrazovce vykonávat. Díky tomuto popisu je znázorněna grafická reprezentace funkcí poskytovaných aplikací. Tyto funkce jsou rozebírány v úvodu kapitoly 7.

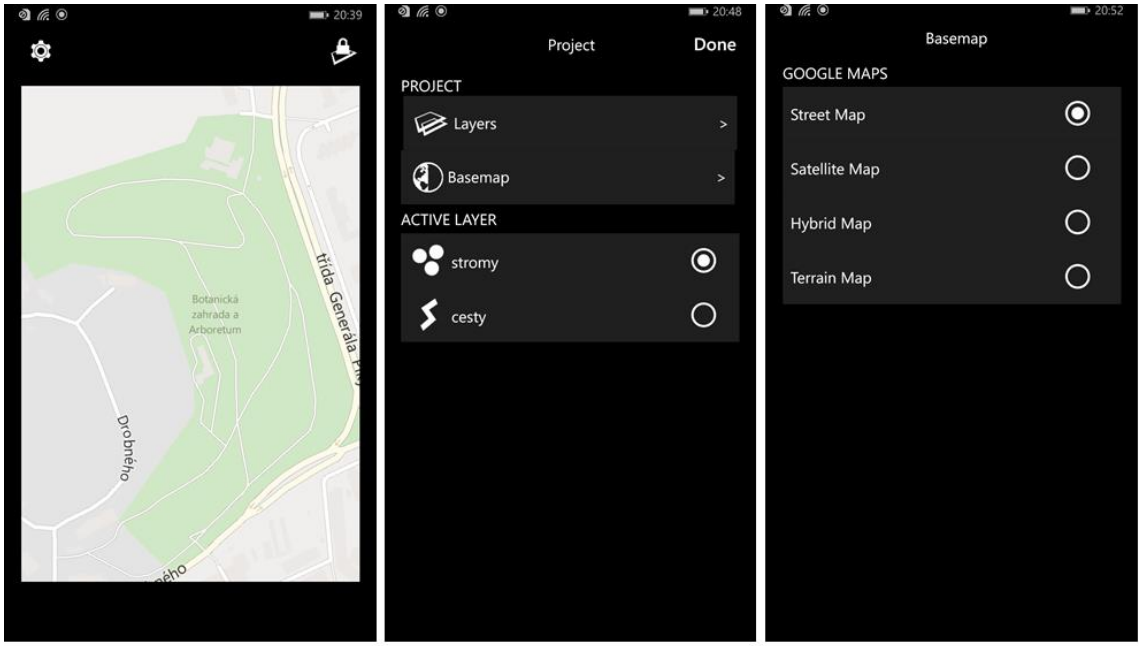


Obr. 14 Obrazovky pro práci s projekty

Po spuštění aplikace se uživateli zobrazí obrazovka pro práci s projekty. Tato obrazovka je opatřena prvkem `Pivot`, který uživatel využije k navigaci mezi obrazovkami využívat gesto `swipe`. Do prvku Pivot jsou přidány dvě obrazovky, které jsou dále rozebírány.

První obrazovka (Obr. 14 – a), která je ihned viditelná po spuštění aplikace slouží k výběru projektu, který bude uživatel používat. Proto je z převážné části obrazovka využita seznamem projektů. Pro každý projekt je zobrazován jeho název a zkrácený popis. V horní liště obrazovky jsou umístěny tlačítka pro přidání nového projektu nebo editaci projektu. Po stisku těchto tlačítek je uživateli zobrazena obrazovka (Obr. 14 – b) opatřená poli pro zadání názvu a popisu projektu. Po stisku tlačítka „Create“ je projekt přidán nebo zeditován, to záleží na uživatelově volbě a automaticky je přikročeno k návratu na předchozí obrazovku. Pokud uživatel kliknutím zvolí určitý projekt, je přikročeno k načítání příslušných vrstev z lokální databáze. Po tuto dobu nemůže uživatel s aplikací nic dělat.

Druhá obrazovka (Obr. 14 – c), zobrazená pomocí `swipe` gesta je využívána uživatelem k práci s projekty. Opět je obrazovka z převážné části využita seznamem projektů. Nyní je ale každý prvek seznamu opatřen zatrhávacím tlačítkem. Ve spodní části obrazovky jsou umístěna tlačítka pro duplikaci a smazání projektu. Po stisku tlačítek jsou zatržené projekty duplikovány nebo odebrány.

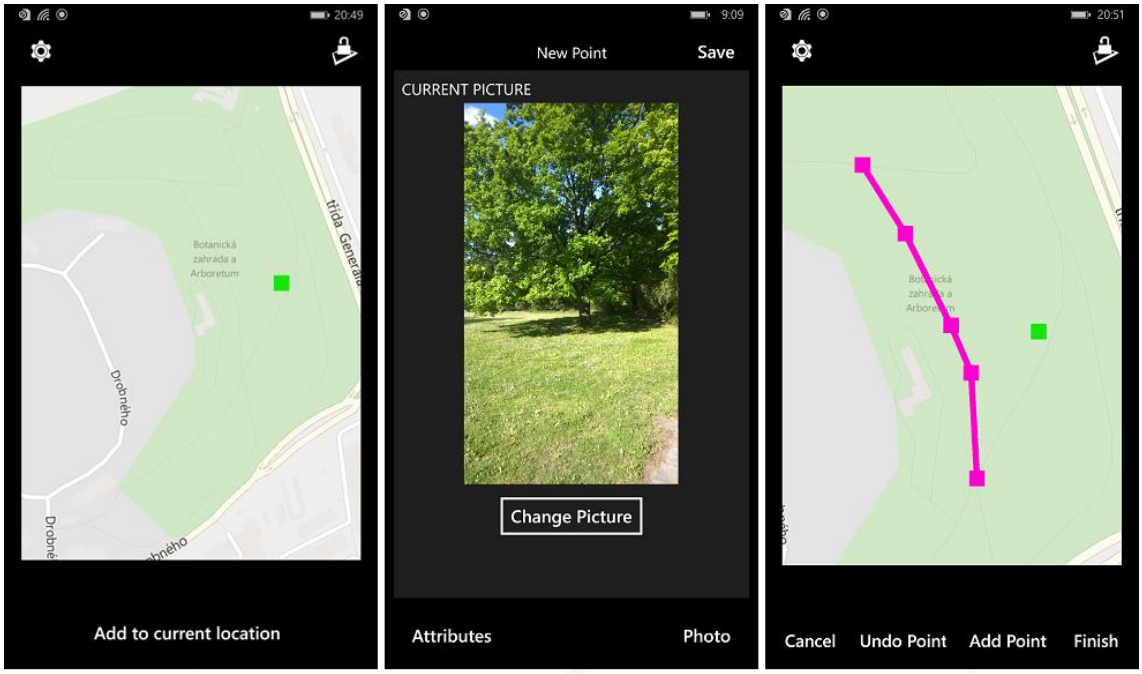


Obr. 15 Hlavní část aplikace pro práci s mapovými prvky a vrstvami

Jakmile jsou do aplikace načteny vrstvy vybraného projektu, je uživateli zobrazena obrazovka umožňující zobrazení mapového prvku (Obr. 15 – a). Proto je opatřena mapou, na kterou jsou mapové prvky zobrazovány. Uživatel pomocí gesta `pinch and zoom` nebo dvojklikem může měnit přiblížení mapy. Dále tahem po mapě může mapu posouvat všemi směry. Pokud uživatel klikne na určitý mapový prvek, může klikem do volného místa na mapě změnit jeho pozici. Dále je v levém horním rohu obrazovky umístěno tlačítko pro přechod do nastavení aplikace (Obr. 15 – b). V pravém horním rohu je umístěno tlačítko pro odemknutí vrstvy a následně je podle typu vrstvy zobrazena obrazovka pro přidání mapového prvku (Obr. 16).

Obrazovka pro nastavení aplikace je rozdělena do dvou částí. Ve spodní části se nachází seznam s viditelnými vrstvami. Pro každou vrstvu je zobrazena ikona demonstrující její typ, název vrstvy a přepínací tlačítko pro výběr aktivní vrstvy. V horní části jsou umístěny prvky pro přechod do správy vrstev a pro zvolení mapového podkladu (Obr. 15 – c). Jako mapový podklad si uživatel může zvolit:

- Terénní mapu
- Hybridní mapu
- Mapu silnic
- Satelitní mapu



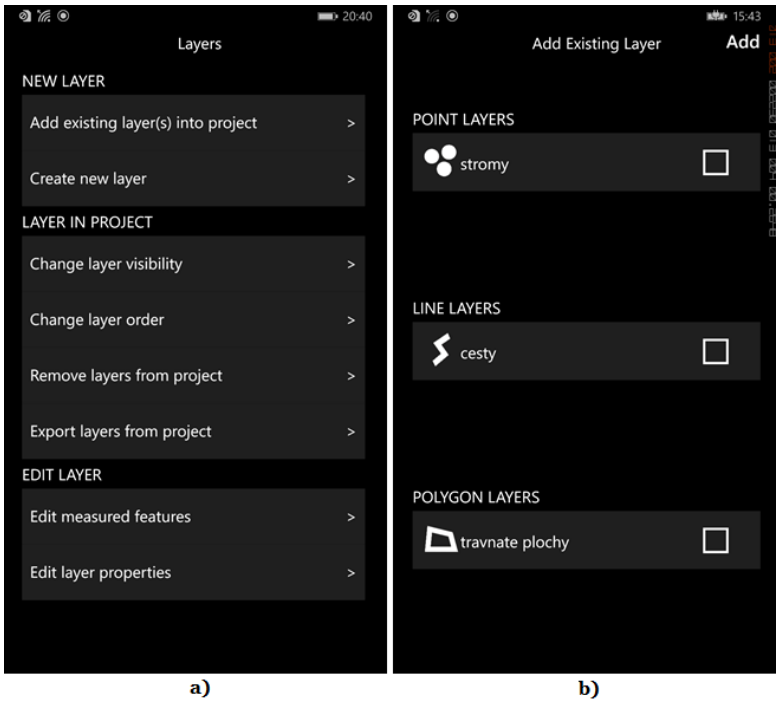
a) b) c)

Obr. 16 Přidání mapového prvku a jeho atributů

Chování aplikace při vytváření mapového prvku bod, polygon nebo line je detailněji popsáno pomocí sekvenčních digramů a diagramů aktivit v kapitole 7.3.2. Jak je uvedeno výše, obrazovky pro přidání mapového prvku se zobrazují v závislosti na typu vrstvy. Pokud je mapový prvek přidáván do bodové vrstvy, je zobrazena obrazovka pro přidání bodu (Obr. 16 – a). Uživatel má opět možnost mapu posouvat, přibližovat a editovat vytvořené mapové prvky akcemi, které jsou popsány u obrazovky (Obr. 15 – a). Dále může uživatel přidat bod dvěma způsoby a to kliknutím do mapy nebo pomocí tlačítka „*Curent postion*“, kdy získá pomocí GPS svoji aktuální pozici. Po zadání souřadnic bodu předchozími způsoby je zobrazena obrazovka pro nastavení vlastností mapového prvku, uvedená v příloze J. Tato obrazovka opět podporuje `swipe` gesto pro přepínání mezi obrazovkami pro zadávání atributů mapového prvku nebo přiřazení fotky mapového prvku (Obr. 16 – b). Pokud chce uživatel přiřadit fotku, musí kliknout na tlačítko „*Change picture*“, kdy se mu následně zobrazí systémová aplikace pro fotoaparát. Po vytvoření fotky je opět zobrazena obrazovka pro přidání vlastností. Po stisku tlačítka „*Save*“ je mapový prvek přidán a automaticky je zobrazena obrazovka pro přidání mapového prvku (Obr. 16 – a).

Pokud je mapový prvek přidáván do liniové nebo polygonové vrstvy, je zobrazena obrazovka pro přidání mapového prvku (Obr. 16 – c). Uživatel má možnost mapu posouvat, přibližovat a editovat vytvořené mapové prvky akcemi, které jsou popsány u obrazovky (Obr. 15 – a). Přidání mapového prvku je provedeno nejprve vytvořením její pozice. Ta je vytvořena postupným kliknutím do mapy, kdy se přidání kontrolního bodu potvrdí tlačítkem „*Add point*“. Také je

možné se v přidávání vracet pomocí tlačítka „*Undo points*“. Přidání pozice je dokončeno stiskem tlačítka „*Finish*“ a je následně zobrazena obrazovka pro přidání vlastností (viz. příloha J), která je popsána výše.



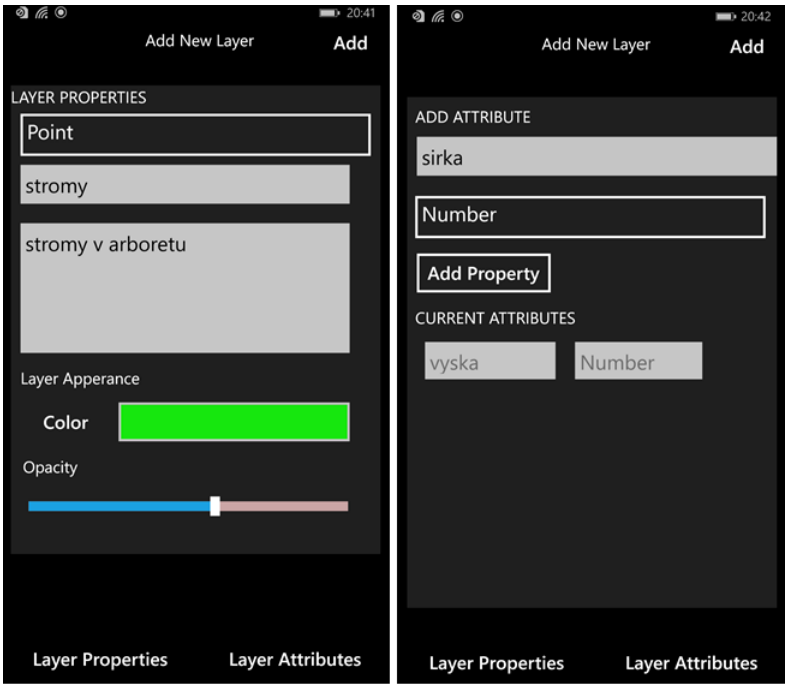
Obr. 17 Menu vrstev a přidání existující vrstvy

Uživatel má možnost pracovat s vrstvami po stisku položky „*Layers*“ v nastavení aplikace (Obr. 15 – b). Následně je uživateli zobrazena obrazovka obsahující menu pro práci s vrstvami a jejich vlastnostmi (Obr. 17 – a). Přidání vrstev může proběhnout dvěma způsoby. To je závislé na uživatelově volbě, zda zvolí možnost „*Add exist layer*“ nebo „*Add new layer*“.

Pokud zvolí možnost „*Add exist layer*“ je zobrazena obrazovka pro přidání existujících vrstev (Obr. 17 – b). Vrstvy jsou roztříděny podle typu do příslušných seznamů. Každá položka (vrstva) v seznamech je opět znázorněna ikonou představující typ vrstvy, jejím názvem a zatrhávacím tlačítkem. Po stisku tlačítka „*Add*“ se zatržené vrstvy přidají do aktuálního projektu.

Pokud uživatel zvolí možnost „*Add new layer*“ je mu zobrazena obrazovka pro vytvoření nové vrstvy (Obr. 18 – a). Obrazovka podporuje `swipe` gesto pro přepínání mezi obrazovkou pro vytvoření vlastností (Obr. 18 – a) a obrazovkou pro vytvoření atributů (Obr. 18 – b). Mezi obrazovkami se lze přepínat také stiskem tlačítek „*Properties*“ a „*Attributes*“. Na obrazovce pro nastavení vlastností může uživatel nastavit např. průhlednost, typ, barva nebo název. Na obrazovce pro vytvoření atributů může uživatel vytvořit atributy s určitým názvem a typem, které jsou po vytvoření zobrazeny ve spodní části obrazovky. Po stisku tlačítka „*Save*“ je vytvořena nová vrstva a automaticky je proveden návrat do menu pro

práci s vrstvami. Chování aplikace při vytváření nové vrstvy je popsáno pomocí diagramu aktivit a sekvenčního a stavového diagramu v kapitole 7.3.1.

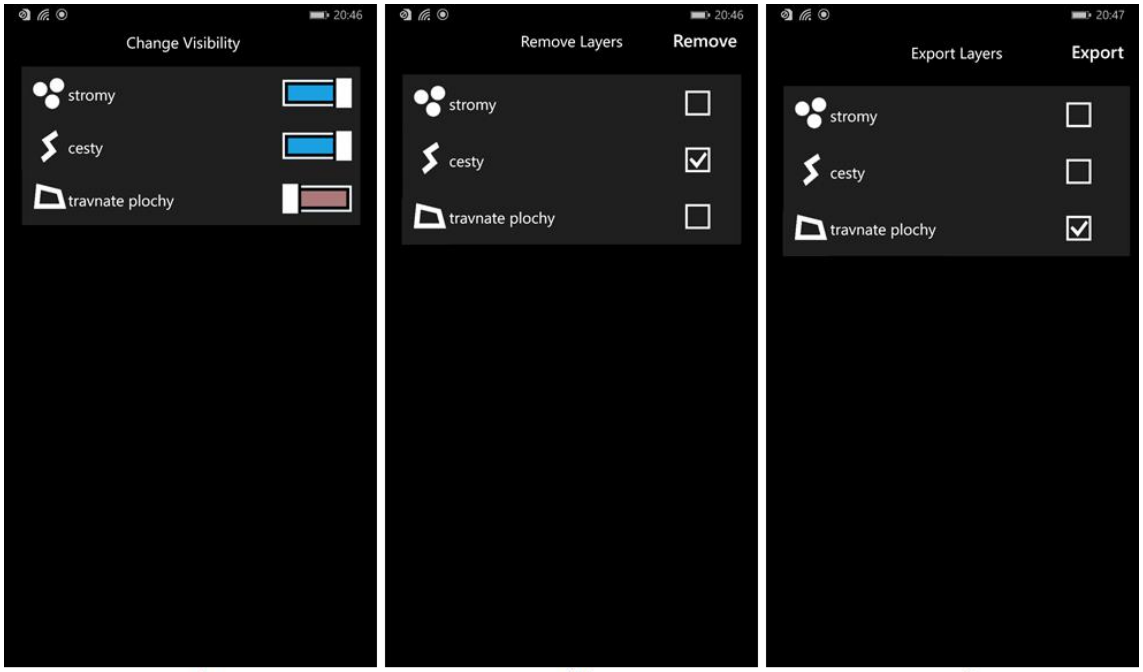


Obr. 18 Přidání nové vrstvy

Po výběru možnosti „Change visibility“ v menu pro práci s vrstvami je zobrazena obrazovka pro změnu viditelnosti (Obr. 19 – a). Převážná část obrazovky je zabrána seznamem vrstev, kde ke každé vrstvě je přidán přepínač. Po stisku přepínače je změna viditelnost vrstvy. To znamená, že mapový prvek vrstvy se budou/nebudou vykreslovat a vrstva bude také přidána/odebrána ze seznamu vrstev na (Obr. 15).

Po výběru možnost „Remove layers“ je zobrazena obrazovka pro odebrání vrstev (Obr. 19 – b). Vrstvy jsou odebrány po stisku tlačítka „Remove“, kdy jsou odebrány zatržené vrstvy.

Po výběru možnosti „Export layers“ je zobrazena obrazovka pro export vrstev do XML souboru (Obr. 19 – c). Zatržené vrstvy jsou exportovány po stisku tlačítka „Export“ a XML soubor je uložen na lokální uložiště telefonu.



Obr. 19 Práce s vrstvami

Po výběru možnosti „*Edit layers*“ v menu pro práci s vrstvami je uživateli zobrazena obrazovka se seznam vrstev (Obr. 20 – a). Jednotlivé vrstvy jsou zobrazeny již známým způsobem (typ vrstvy a její název). Po výběru vrstvy je zobrazena obrazovka pro editaci vrstvy (Obr. 18 – a), která byla popisována výše ve spojitosti s přidáním vrstvy. Rozdíl v zobrazování obrazovky je takový, že v případě editace vrstvy jsou prvky GUI předvyplněny vlastnostmi vrstvy.

Po výběru možnosti „*Edit measured feature*“ je opět uživateli zobrazena obrazovka se seznam vrstev (Obr. 20 – a). Po výběru určitě vrstvy jsou uživateli zobrazeny jednotlivé mapové prvky. Pro každý mapový prvek je zobrazeno jeho id, vybraný atribut a tlačítka „*Delete*“ a „*Edit*“ pro smazání a editaci mapového prvku. Po stisku tlačítka pro editace je zobrazena obrazovka pro editaci atributů mapového prvku (Obr. 20 – b). Tato obrazovka opět podporuje `swipe` gesto. Uživatel může na obrazovce upravovat atributy mapového prvku nebo informace o datu vytvoření a poslední editace mapového prvku. Po vykonání `swipe` gesta je zobrazena obrazovka pro úpravu obrázku (Obr. 16 – c), připojeného k mapovému prvku. Uživatel může obrázek změnit kliknutím na tlačítko „*Change picture*“ kdy je poté spuštěna systémová aplikace pro fotoaparát. Po případném vytvoření fotky a návratu zpět je fotka uživateli zobrazena. Proces editace je ukončen tlačítkem „*Save*“ umístěném v pravém horním rohu. Mezi obrazovkami se lze přepínat také pomocí tlačítek „*Attributes*“ a „*Photo*“.

Uživatel má také možnost vícepoložkového odebírání mapových prvků. To je možné po stisku tlačítka „Edit“ na obrazovce se seznamem mapových prvků (Obr. 20 – c). Následně je u jednotlivých mapových prvků zobrazeno zatrhávací

tlačítko. Uživatel má možnost označit všechny mapové prvky pomocí tlačítka „*Select all*“. Po stisku tlačítka „*Remove*“ v pravém horním rohu je zobrazeno dialogové okno s upozorněním na vícepoložkové mazání. Po potvrzení stiskem tlačítkem „*Remove*“ v dialogovém okně je smazání položek dokončeno. Uživatel může mazání mapových prvků zrušit stiskem tlačítka „*Cancel*“ v dialogovém okně.

Obr. 20 Editace vlastností mapových prvků

8 Testování

Než mohla být aplikace prohlášena za správně fungující, musela být nejprve důkladně otestována. Testování aplikace probíhalo ve dvou krocích. Tyto kroky byly rozděleny podle požadovaných kritérií na daný test. V prvním kroku probíhalo offline testování pomocí emulátoru, který je zabudován přímo ve vývojovém prostředí. V druhém kroku probíhalo testování pomocí zařízení Nokia Lumia 1520, s nainstalovaným operačním systém Windows Phone 8. Tyto dva kroky testování jsou dále detailněji rozepsány.

8.1 Testování pomocí emulátoru

K testování pomocí emulátoru bylo přikročeno v průběhu vývoje aplikace. Proto tento krok testování měl za úkol poukázat na nedostatky v rámci:

- Funkční logiky aplikace
- Vytvořeného GUI

V rámci funkční logiky byly testovány jednotlivé funkce aplikace. Dle analytického modelu tříd uvedeného v příloze D byly prováděny testy databázového modulu, modulu pro HTTP komunikaci se serverem a modulu pro reprezentaci dat v aplikaci. Tyto testy byly prováděny za účelem ověření správné funkčnosti aplikace, zejména aby nenastalo jakékoliv neočekáváné ukončení aplikace.

Dále byly prováděny testy vytvořeného GUI. Bylo testováno, zda vytvořené jednotlivé stránky, tvořící GUI aplikace odpovídají předloze multiplatformního návrhu GUI. Dále bylo testováno, zda přechody mezi stránkami, které jsou uskutečněny po uživatelových akcích, odpovídají přechodům navrženým v multiplatformním návrhu GUI. Následně bylo testováno propojení GUI s funkční vrstvou aplikace. Tyto testy měly za úkol ověřit správnost plnění GUI daty z funkční vrstvy. Dále měly ověřit způsob manipulace s prvky GUI, především s mapou.

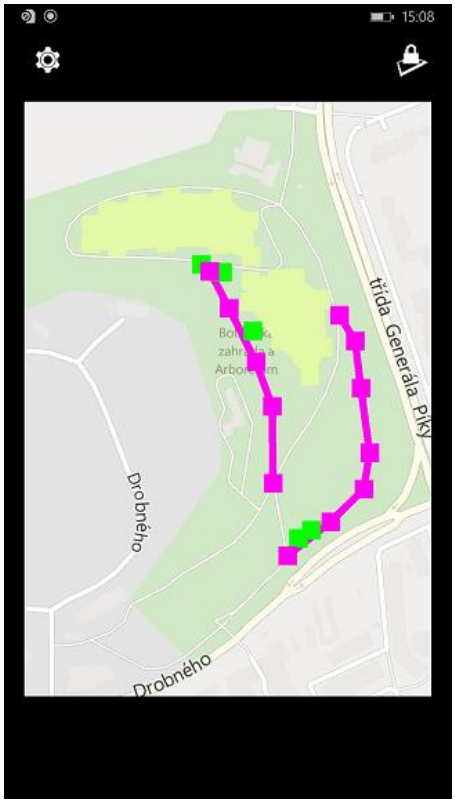
Test manipulace s prvky GUI odhalil například nemožnost použití dvojkliku při výběru mapového prvku. To z toho důvodu, že akce dvojkliku na mapě je použita pro přibližování mapy. Testování pomocí emulátoru neprokázalo další problémy. Proto následující části testu GUI byly uskutečněny pomocí mobilního zařízení. Test HTTP komunikace a přechodů mezi stránkami neodhalil žádné nedostatky.

8.2 Testování pomocí mobilního zařízení

K testování pomocí mobilního zařízení bylo přikročeno na samý závěr vývoje aplikace. Tento krok měl za úkol ověřit správnou funkčnost akcí GUI, které nebylo možné ověřit pomocí emulátoru. Dále použitelnost vytvořených funkcí na mobilním zařízení a jejich uživatelskou přívětivost.

Nejprve bylo přikročeno k dokončení testu použitelnosti GUI na mobilním zařízení. Testovány byly postupně gesta pro práci s mapou, tedy především gesto `pinch and zoom` použité pro změnu přiblížení mapy. Dále bylo zkoumáno počáteční přiblížení mapy. Z testu emulátoru byla použita hodnota přiblížení 10, která odpovídala přiblížení 152,87 metru na pixel. Tato hodnota se ukázala jako nevyhovující a byla stanovena hodnota přiblížení 18, která odpovídá přiblížení 0,60 metru na pixel.

Následně byl znovu proveden test přechodů mezi stránkami aplikace. Tento test měl především poukázat na použitelnost tlačítka návratu na předchozí stránku (umístěno v levém horním rohu), které lze vidět v příloze A na stránce s typem mapového podkladu. Tyto tlačítka se ukázala jako nepoužitelná z důvodu velké vzdálenosti, kterou by musel uživatel ke kliku urazit. Tlačítka byla z aplikace odstraněna a pro návrat zpět je použito pouze systémové tlačítko pro návrat.



Obr. 21 Testování aplikace

Dále byl proveden test použitelnosti vytvořených funkcí. Test byl proveden v arboretu Mendelovy univerzity v Brně. Nejprve byl pomocí stránky na Obr. 14 vytvořen projekt, do kterého byly postupně pomocí stránek uvedených na Obr. 18 vytvořeny bodová vrstva stromy, liniová vrstva cesty a polygonová vrstva travnaté plochy. Do vytvořených vrstev byly pomocí stránek na Obr. 16 do vrstev přidány

mapové prvky. Výsledek testu je zobrazen na Obr. 21, kde stromy (body) jsou zobrazeny zeleně, cesty (linie) fialově a travnaté plochy (polygony) žlutě. Test ukázal, že vytvořené funkce jsou použitelné pro práci s aplikací.

Závěrečný test proběhl pomocí poskytnutí aplikace testovacímu subjektu. Ten měl za úkol pomocí aplikace založit projekt, následně do projektu přidat bodovou vrstvu a do té přidat mapový prvek. Následně mapový prvek smazat.

Po úvodním seznámení a instruktáži s aplikací byl testovaný subjekt schopen zadaný úkol splnit. Při práci na úkolu testovaný subjekt poskytl užitečné postřehy týkající se akcí, které v aplikaci chyběly. Postřehy byly následně do aplikace implementovány. Těmito postřehy byly:

- Automatické přechody zpět po vytvoření nebo editaci vrstvy
- Grafické znázornění při duplicitním názvu vrstvy

Dále poskytl postřehy týkající se programátorských nedostatků v aplikaci, které byly následně odstraněny. Jednalo se zejména o:

- Špatné vykreslování mapových prvků při jejich editaci,
- Možnost kliku do pole, které slouží pouze pro znázornění barvy vrstvy,
- Zadání duplicitního názvu projektu.

Na základě provedených testů byly odstraněny výše uvedené nedostatky a vedly k ověření správné funkčnosti aplikace.

9 Diskuze a závěr

9.1 Shrnutí

Práce se zabývá problematikou vývoje GIS aplikace pro platformu mobilního operačního systému Windows Phone 8. Pro implementaci aplikace byl zvolen programovací jazyk C# s využitím technologie Windows Presentation Foundation pro tvorbu GUI a externích knihoven pro práci s databází nebo JSON soubory.

První kapitola obsahuje souhrn termínů používaných v práci. Pojmy byly většinou uváděny ve zkratkách. Z toho důvodu je ke každé zkratce podáno vysvětlení ve formě jejího celého názvu, viz kapitola 1 Seznam zkratek a vybraných pojmů.

Druhá kapitola uvádí čtenáře do problematiky GIS aplikací. V kapitole jsou postupně vysvětleny základní pojmy problematiky GIS, jako jsou např. Geografický informační systém, kartografický přístup ke GIS nebo základní mapové prvky (bod, polygon, linie). Dále jsou v kapitole stanoveny základní cíle a podněty, které vedly k vytvoření aplikace. Více viz kapitola 2 Úvod a cíl práce.

Třetí kapitola představuje hlavní teoretickou část práce. Obsahuje aktuální stav řešené problematiky, tedy GIS aplikace, které jsou již pro mobilní platformy vytvořeny, především aplikace ArcGis a QField. Dále obsahuje detailní popis platformy Microsoft Windows Phone 8 z pohledu vývojáře. Je tedy rozebírána architektura platformy a SDK použité k vývoji aplikace. Dále kapitolo obsahuje popis částí frameworku .NET, které byly použity pro návrh GIS aplikace. Jedná se především o části komunikace s databází, podpory HTTP komunikace a tvorby GUI. Více o této problematice viz kapitola 3 Rešeršní část.

Čtvrtá kapitola pojednává o postupu řešení při vývoji GIS aplikace. Kapitola je rozdělena do sekcí, které vždy pojednávají o postupu vývoje dané sekce, např. lokální databáze, funkční vrstvy nebo GUI. Kapitola osahuje popis funkcí, které sekce bude poskytovat, co bude předlouhou pro vytvoření dané sekce apod. Více viz kapitola 4 Metodika.

Pátá kapitola rozebírá požadavky, které musí vytvořená aplikace splňovat. Kapitola je rozdělena na požadavky stanovené zadavatelem a na omezení vyplívající z multiplatformního vývoje. Požadavky stanovené zadavatelem především určují, jaké funkce má aplikace poskytovat. Omezení vyplývající z multiplatformního vývoje určují základní logiku aplikace. Je zde určen vzhled GUI aplikace, způsob komunikace se serverem a způsob řešení synchronizace dat. Více viz kapitola 5 Požadavky na aplikaci.

Šestá kapitola konkretizuje nejasné otázky vyplývající z nejednoznačnosti základního cíle práce. Konkretizuje otázky, jako jsou: „Jak reprezentovat prvky vrstev?“, „Jaká funkcionalita bude poskytována pro práci s mapovými prvky“ a „Jaké požadavky zákazníka mají být splněny?“. Více viz kapitola 6 Konkretizace cíle práce.

Sedmá kapitola pojednává o vlastním vývoji aplikace. Postupně je rozebrán způsob návrhu a implementace databázové a funkční vrstvy. Dále kapitola obsahuje modely upřesňující chování vybraných částí aplikace. Dále obsahuje popis

GUI aplikace z uživatelského hlediska, tedy jaké akce uživatel může na dané stránce vykonávat, jakým způsobem probíhá využití dané funkcionality aplikace a způsob navigace mezi stránky. Více o této problematice viz kapitola 7 Návrh a implementace aplikace.

Osmá kapitola se zabývá testováním aplikace a to jak z pohledu offline testování pomocí emulátoru tak i z pohledu online testování pomocí mobilního zařízení Nokia Lumia 1520. Aplikace také byla testována pomocí testovacího subjektu a jeho připomínky vedly k vylepšení nedostatků aplikace. Více viz kapitola 8 Testování.

9.2 Diskuze

Vytvořená aplikace je určena k vytváření map v oblasti GIS. K této činnosti může uživatel použít veškerou požadovanou funkcionalitu. Vytvořená aplikace se jeví i neobeznámeným uživatelům jako intuitivní a jednoduchá pro použití v terénu.

Během vývoje aplikace se vyskytlo několik překážek komplikujících vývoj aplikace. Ten nejzávažnější problém nastal při návrhu lokální databáze. Jednalo se o problém s kompatibilitou verzí formátu SDF použitého pro vytvoření databáze. Na základě milných informací na webových stránkách MSDN byl nejprve použit databázový soubor SDF verze 4.0. Následný vývoj a prohledání dalších informačních zdrojů ukázalo, že je nutné použít nižší verzi a to 3.5. Dalším problémem bylo umístění lokální databáze do složky aplikace, umístěné na „*isolated storage*“ telefonu. Tento problém byl odstraněn umístěním lokální databáze do instalační složky aplikace.

V aktuální verzi je zpracována jednostranná komunikace mezi aplikací a serverem. To je dáno implementací na straně mapového serveru, který nebyl v době odevzdání diplomové práce zcela dokončen. Autor práce počítá s dalším rozšířením aplikace a to v podobě oboustranné komunikace.

9.3 Závěr

Cílem diplomové práce bylo navrhnout a následně implementovat GIS aplikaci pro platformu operačního systému Microsoft Windows Phone 8.

Cíl práce byl splněn, koncový uživatel má k dispozici plnohodnotnou GIS aplikace, která splňuje stanovené požadavky. Funkčnost navrženého řešení byla prokázána provedením jak simulačních tak i reálných testů pomocí mobilního zařízení.

10Literatura

.NET API for Windows Phone. Windows Dev Center [online]. © 2015 [cit. 2015-04-28]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/apps/jj207211(v=vs.105).aspx

ArcGIS - Add a new feature. ArcGIS [online]. © 2014 [cit. 2015-04-14]. Dostupné z: http://doc.arcgis.com/en/arcgis-app/windows-phone/help/add-a-new-feature.htm

ArcGIS - System requirements. ArcGIS [online]. © 2015 [cit. 2015-04-14]. Dostupné z: http://doc.arcgis.com/en/arcgis-app/

ArcGIS - Use the app. ArcGIS [online]. © 2015 [cit. 2015-04-14]. Dostupné z: http://doc.arcgis.com/en/arcgis-app/windows-phone/help/use-the-app-faqs.htm

ArcGIS for Windows Phone overview. ArcGIS [online]. © 2014 [cit. 2015-04-14]. Dostupné z: http://doc.arcgis.com/en/arcgis-app/windows-phone/help/arcgis-for-windows-phone-overview.htm

Asynchronous Programming with Async and Await (C# and Visual Basic). MSDN – Microsoft Developer Network< [online]. © 2015 [cit. 2015-04-21]. Dostupné z: https://msdn.microsoft.com/en-us/library/hh191443.aspx

HSU, L S. -- OBE, R O. *PostGIS in Action, Second Edition*. USA: Manning Publications Co, 2014. 625 s. ISBN 978-1-6172-9139-5.

HttpClient Class. MSDN – Microsoft Developer Network< [online]. © 2015 [cit. 2015-04-21]. Dostupné z: https://msdn.microsoft.com/en-us/library/system.net.http.httpclient%28v=vs.118%29.aspx

Json.NET Documentation. Json.NET - Newtonsoft [online]. © 2015 [cit. 2015-04-21]. Dostupné z:http://www.newtonsoft.com/json/help/html/Introduction.htm

Local database for Windows Phone 8. Windows Dev Center [online]. © 2015 [cit. 2015-04-21]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/apps/hh202860%28v=vs.105%29.aspx

MACDONALD, Matthew a Mario SZPUSZTA. 2008. ASP.NET 3.5 a C# 2008: tvorba dynamických stránek profesionálně. Vyd. 1. Brno: Zoner Press, 1584 s. ISBN 978-80-7413-008-3.

MCKEANA, S. -- WHITECHAPEL, A. *Windows Phone 8 Developmental Internals*. United States: Microsoft Press,U.S., 2013. 1044 s. ISBN 978-0-7356-7623-7.

Návrh GUI pro platformu iOS. Brno, 2014.

PETZOLD, C. *Programming Windows Phone 7*. Redmond: Microsoft Press, 2010. 997 s. ISBN 978-0-7356-4335-2.

QField. OpenGIS [online]. © 2015 [cit. 2015-04-14]. Dostupné z: http://www.opengis.ch/android-gis/qfield/

System requirements for the emulator for Windows Phone 8. Windows Dev Center [online]. © 2015 [cit. 2015-04-14]. Dostupné z: https://msdn.microsoft.com/cs-cz/library/windows/apps/ff626524(v=vs.105).aspx

The Windows Phone Toolkit. CodePlex - Open Source Project Hosting [online]. 15.8.2013 [cit. 2015-04-22]. Dostupné z: http://phone.codeplex.com/

Úvod do Geografických informačních systémů. Geomatika [online]. 2005 [cit. 2015-04-14]. Dostupné z: http://gis.zcu.cz/studium/ugi/e-skripta/ugi.pdf

VAUGHAN, D. *Windows Phone 8 Unleashed*. United States: Sams Publishing, 2013. 65 s. ISBN 978-0-672-33689-8.

WILDERMUTH, S. *Essential Windows Phone 8*. United States: Addison-Wesley Educational Publishers Inc, 2013. ISBN 978-0-321-90494-2.

Win32 and COM API for Windows Phone 8. Windows Dev Center [online]. © 2015 [cit. 2015-04-28]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/apps/jj207198(v=vs.105).aspx

Windows Phone API reference. Windows Dev Center [online]. © 2015 [cit. 2015-04-28]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/apps/ff626516%28v=vs.105%29.aspx

Windows Phone Emulator for Windows Phone 8. Windows Dev Center [online]. © 2015 [cit. 2015-04-14]. Dostupné z: https://msdn.microsoft.com/library/windows/apps/ff402563(v=vs.105).aspx

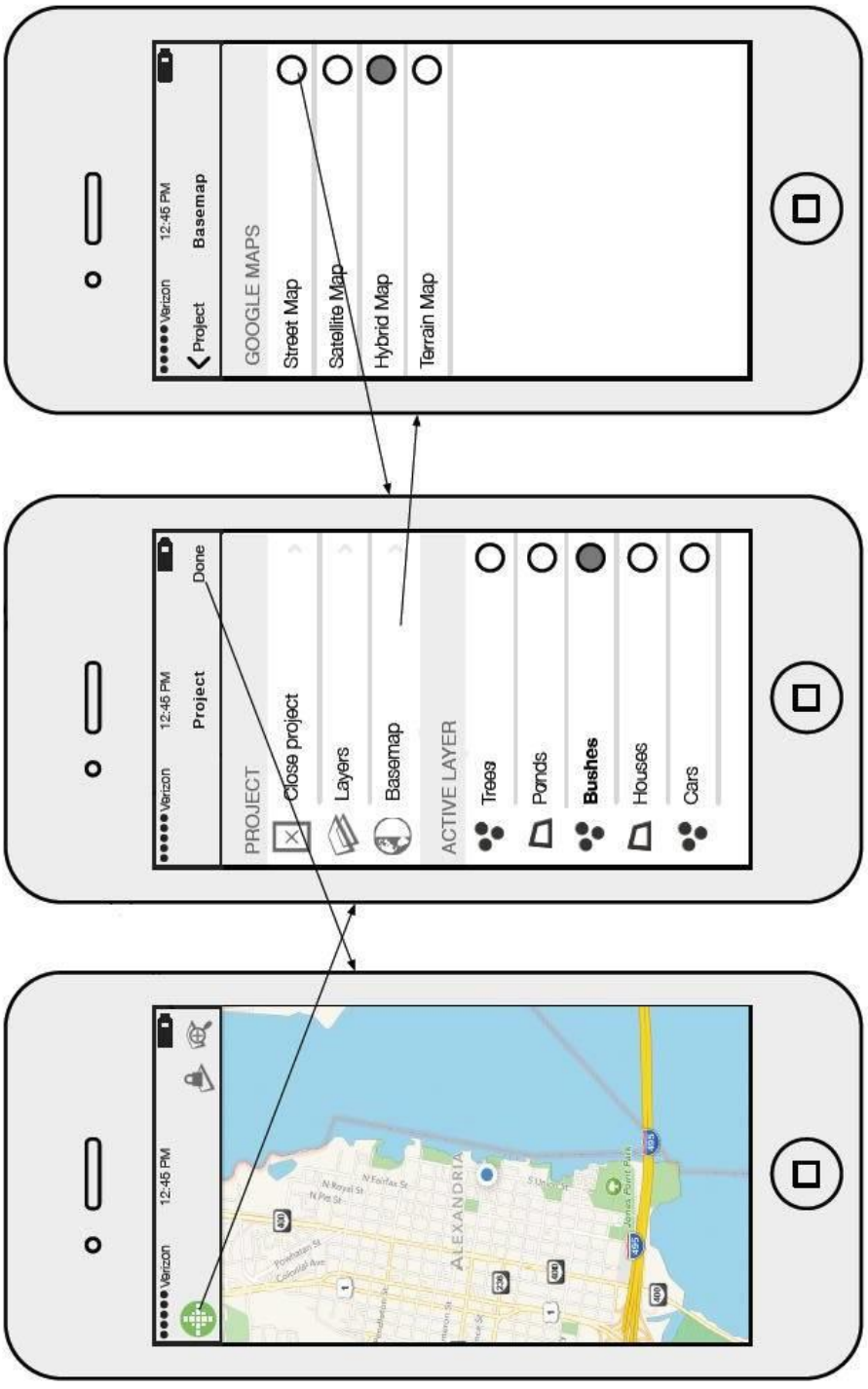
Windows Phone Runtime API. Windows Dev Center [online]. © 2015 [cit. 2015-04-28]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/apps/jj207212(v=vs.105).aspx

XAML Namespaces and Namespace Mapping for WPF XAML. MSDN-the microsoft developer network [online]. © 2015 [cit. 2015-04-21]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms747086(v=vs.110).aspx

XAML Overview (WPF). MSDN-the microsoft developer network [online]. © 2015 [cit. 2015-04-21]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms752059(v=vs.110).aspx

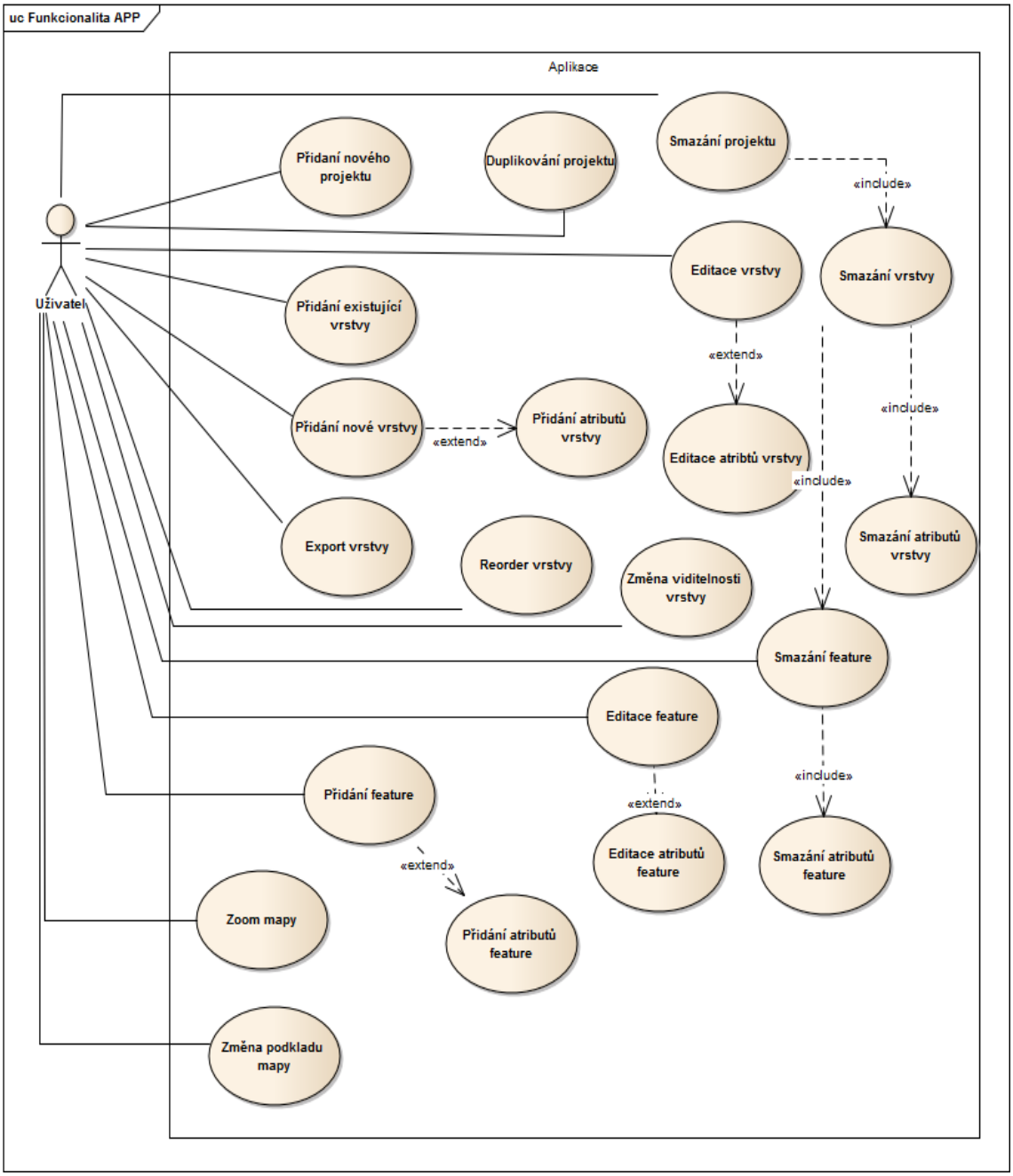
Přílohy

A Návrh GUI



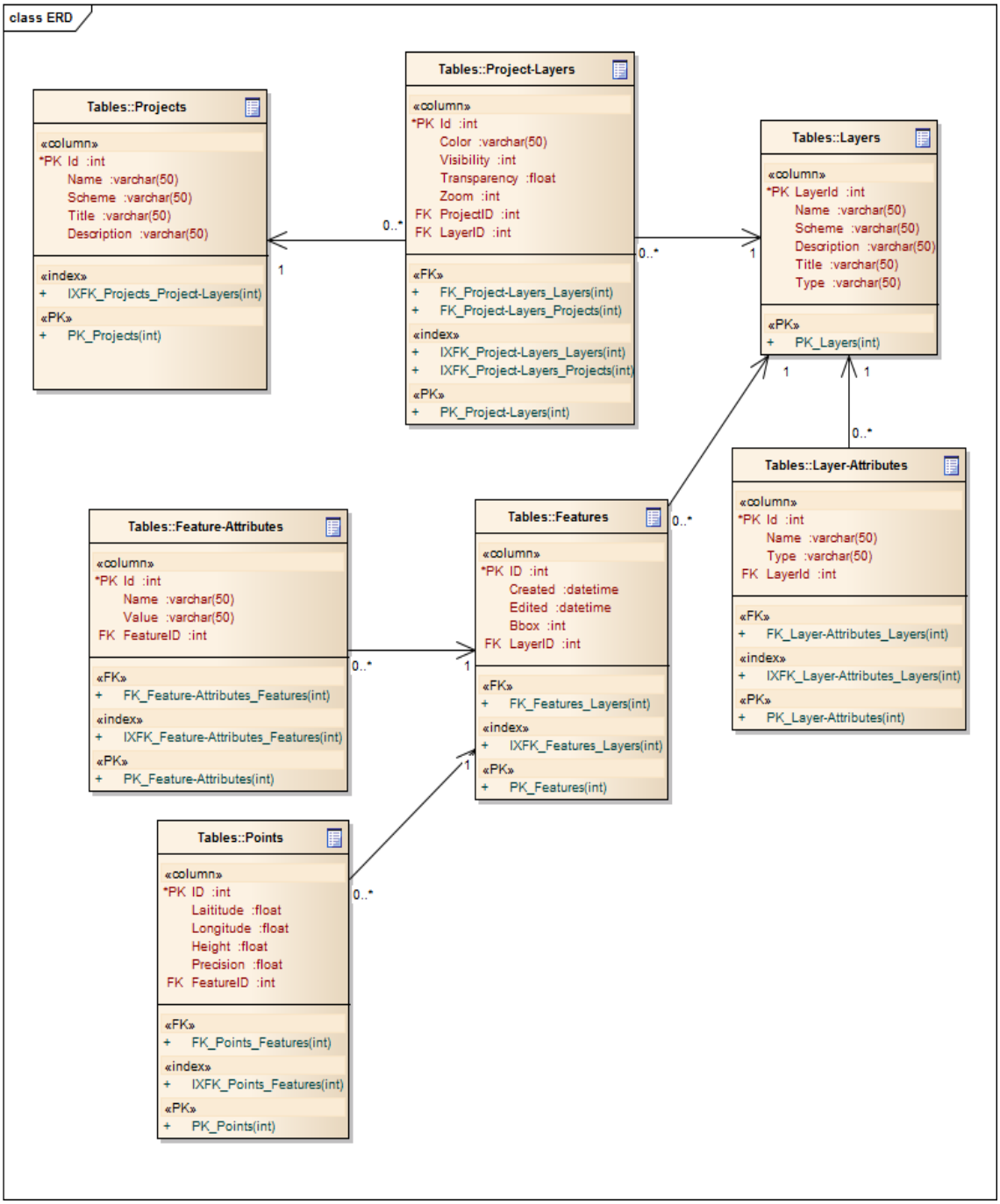
Obr. 22 Návrh GUI
Zdroj: (Návrh GUI pro platformu iOS, 2014)

B Use Case diagram



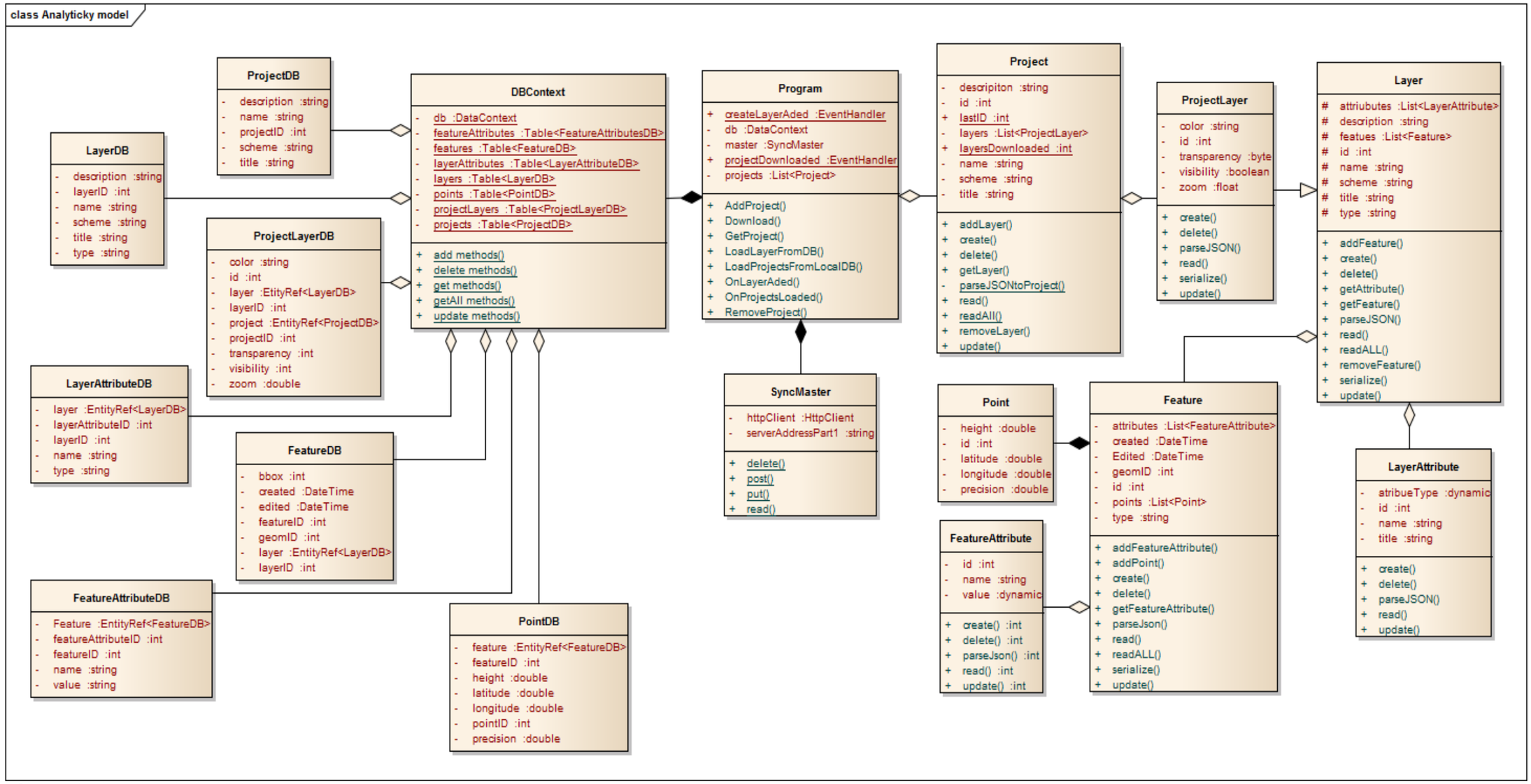
Obr. 23 Funkcionalita aplikace

C ERD lokální databáze



Obr. 24 ERD diagram lokální databáze

D Analytický model tříd



Obr. 25 Analytický model funkční vrstvy

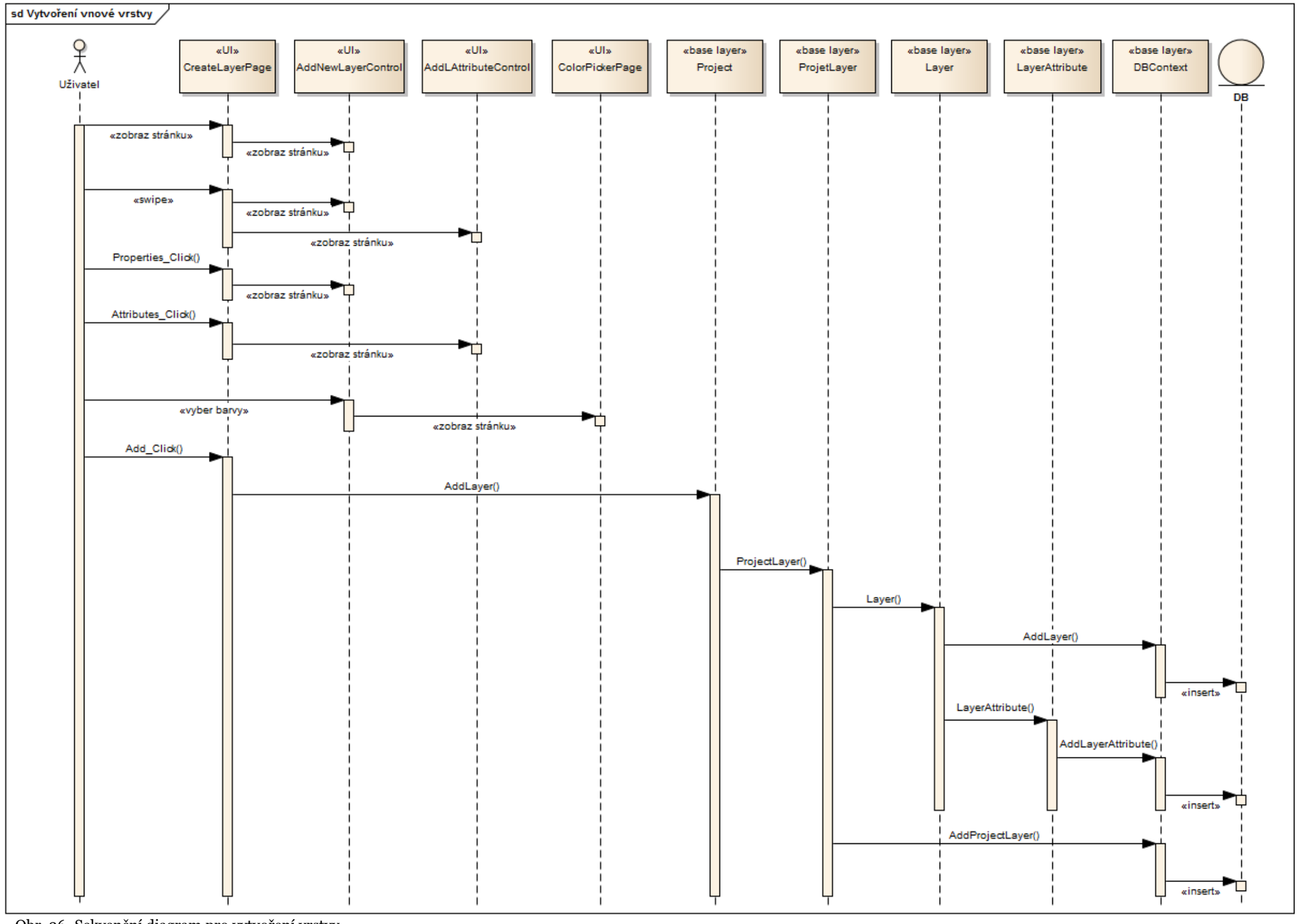
E Datová třída ProjectLayerDB

```
[Table(Name = "ProjectLayers")]
    class ProjectLayerDB
    {
        private int id;
        [Column(IsPrimaryKey = true, Name = "ProjectLayerID")]
        public int ID { get { return id; } set { id = value; } }

        private double zoom;
        [Column(Name = "Zoom")]
        public double Zoom { get { return zoom; } set { zoom = value; } }

        private int visibility;
        [Column(Name = "Visibility")]
        public int Visibility { get{return visibility;} set{visibility = value;}}

        private int transparency;
        [Column(Name = "Transparency")]
        public int Transparency { get { return transparency; } set { transparency
= value; } }

        private string color;
        [Column(Name = "Color")]
        public string Color { get { return color; } set { color = value; } }

        private int projectID;
        [Column(Name = "ProjectID")]
        public int ProjectID { get { return projectID; } set { projectID = value;
} }

        private int layerID;
        [Column(Name = "LayerID")]
        public int LayerID { get { return layerID; } set { layerID = value; } }

        private EntityRef<ProjectDB> project;
        [Association(IsForeignKey = true, ThisKey = "ProjectID", OtherKey = "Pro-
jectID", Storage = "project")]
        public ProjectDB Project { get { return project.Entity; } }

        private EntityRef<LayerDB> layer;
        [Association(IsForeignKey = true, ThisKey = "LayerID", OtherKey = "Lay-
erID", Storage = "layer")]
        public LayerDB Layer { get { return layer.Entity; } }
    }
```

F Stahování dat ze serveru

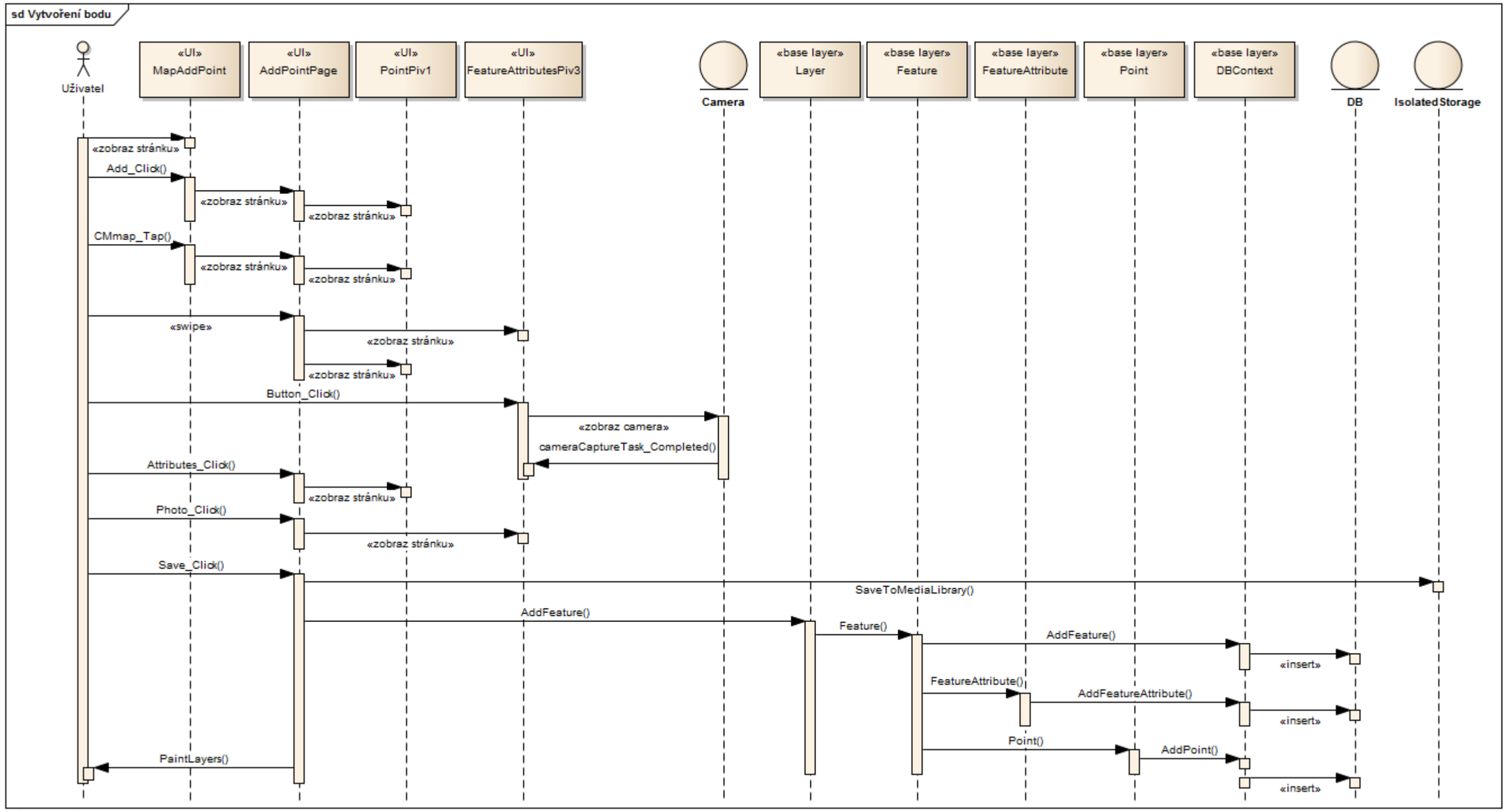
```
public static async Task<String> get(string part2)
        {
            Uri resourceUri;
            if (!Uri.TryCreate(httpClient.BaseAddress.AbsoluteUri + part2, Uri-
Kind.Absolute, out resourceUri))
            {
                return null;
            }
            if (resourceUri.Scheme != "http" && resourceUri.Scheme != "https")
            {
                return null;
            }

            try
            {
                string result = await httpClient.GetStringAsync(resourceUri);
                return result;
            }
            catch (Exception ex)
            {
                return null;
            }
        }
```

G Sekvenční diagram pro vytvoření vrstvy



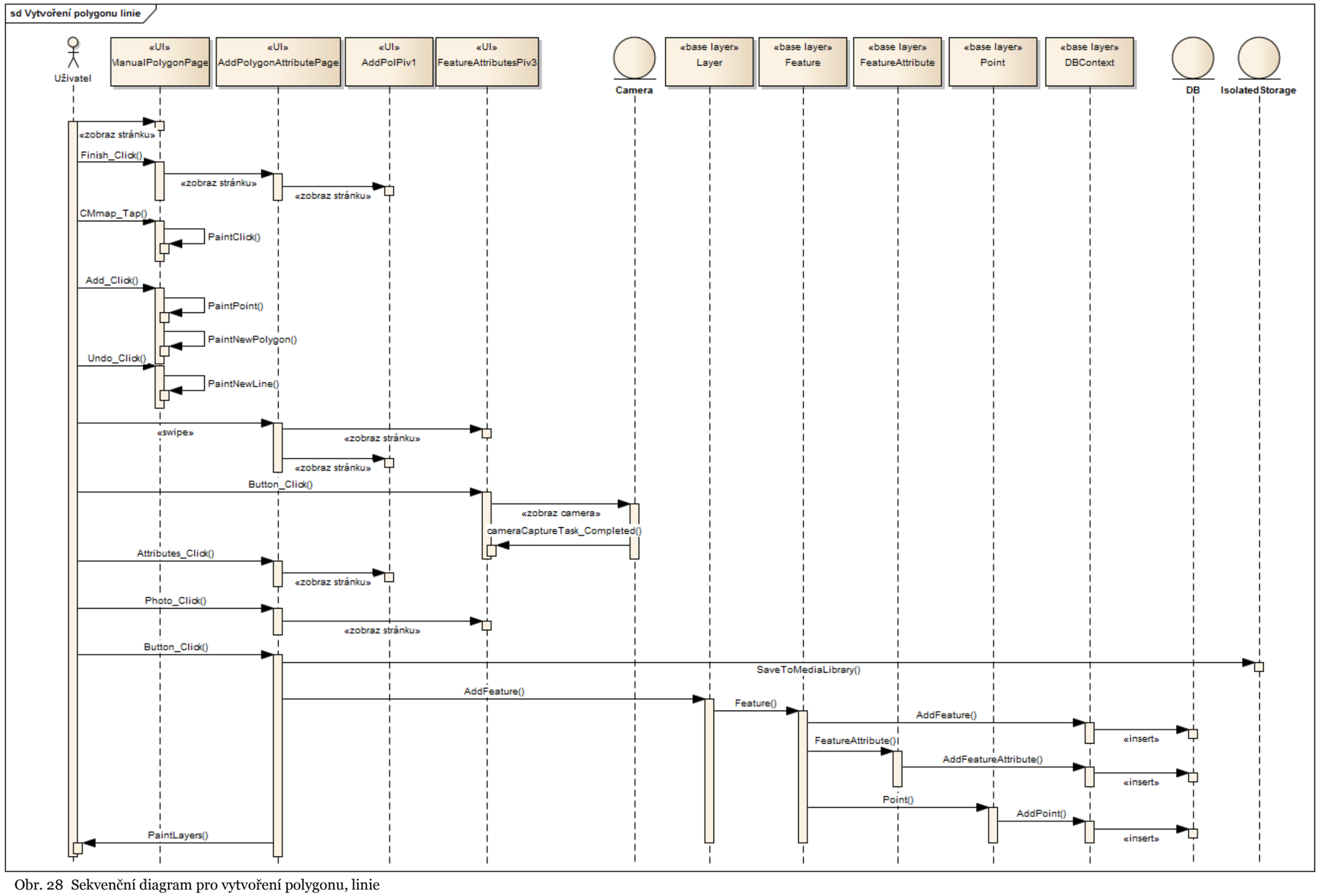
Obr. 26 Sekvenční diagram pro vytvoření vrstvy

H Sekvenční diagram pro vytvoření bodu



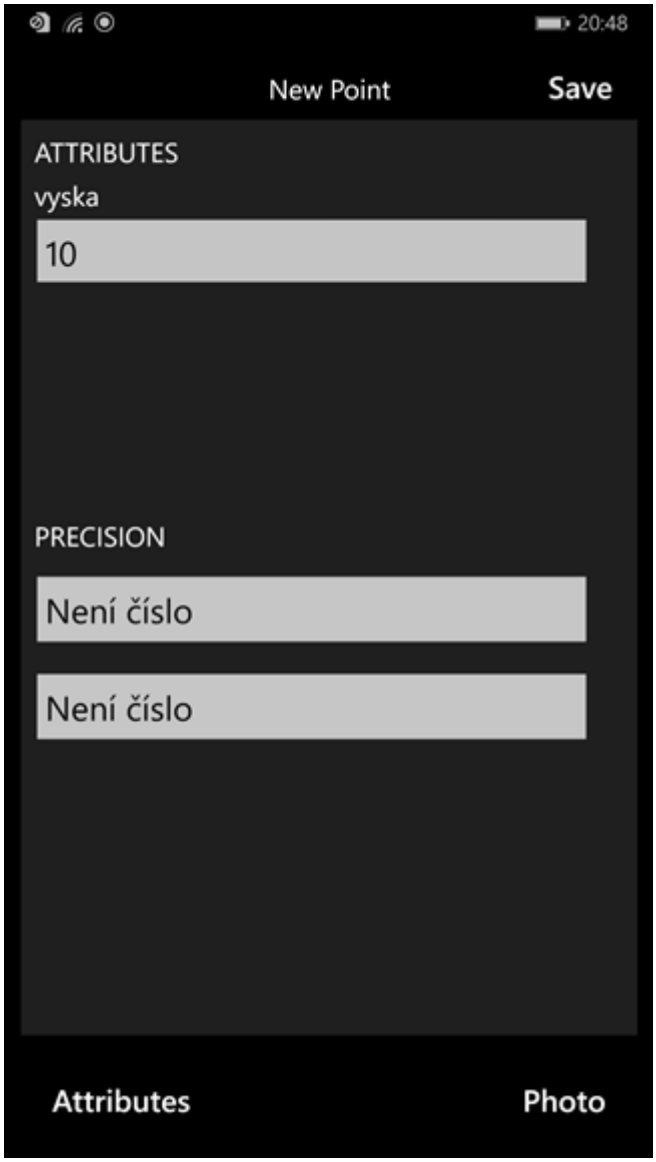
Obr. 27 Sekvenční diagram pro vytvoření bodu

I Sekvenční diagram pro vytvoření polygonu nebo linie



Obr. 28 Sekvenční diagram pro vytvoření polygonu, linie

J Nastavení vlastností vrstvy



Obr. 29 Stránka pro nastavení vlastnosti vrstvy